Les 20EMES journées de
L'Association Française d'Informatique Graphique

AFIG 2007

26 - 28 novembre 2007

Marne-La-Vallée

îledeFrance

GASPARD MONGE INSTITUT

imac

AFIG

GROUPE ESIEE PARIS

UNIVERSITÉ PARIS-EST Marne-la-Vallée

20$^{\text{èmes}}$ journées de l'Association Française
d'Informatique Graphique 2007
AFIG 2007
26-28 novembre 2007

Edito

L'université Paris-Est est fière d'acceuillir cette année les 20èmes journées de l'Association Française d'Informatique Graphique. Il s'agit d'un rendez-vous d'exception pour lequel un bilan de ces vingt dernières années va être dressé. C'est l'occasion pour la communauté de revoir le passé pour mieux définir l'avenir. Modélisation, techniques de rendu, simulation de phénomènes naturels, visualisation scientifique ou encore réalité virtuelle sont des exemples de domaine représenté cette années dans nos journées.

Je souhaite profiter de la tenue de la conférence dans notre région, pour associer, cette année, le monde entrepreneurial à la manifestation. En effet de nombreuses sociétés françaises sont à la pointe de leurs domaines et des liens étroits méritent d'être tissés entre l'excellence de nos équipes de recherche et la qualité de ces entreprises. Cette année, les entreprises de la post-production numérique ainsi que des jeux vidéo ont répondu à l'appel et participent activement aux journées.

Comme toujours, les journées de l'AFIG restent un évènement incontournable pour nos doctorants. Elles leur permetent de présenter leurs travaux à la communauté scientifique et de rencontrer les chercheurs français des autres laboratoires. Nous avons donc tenu à favoriser leur participation active aux journées.

Pour conclure, je veux adresser de chaleureux remerciements à toutes les personnes m'ayant aidé dans la bonne réalisation de ces journées :

- Patrice Bouvier, François de Sorbier et Benjamin Raynal pour leur véritable dévouement à l'organisation de cet évènement,
- Line Fonfrède pour sa profonde connaissance des arcanes administratives et financières de l'université,
- Michel Couprié et Eric Incerti pour leur aide précieuse et leurs talents organisationnels,
- Jean Pierre Jessel, Daniel Menevaux, Vincent Boyer, Dominique Bechmann, et les membres du CA de l'AFIG, pour leur soutien et leurs précieux conseils dans l'organisation des journées,
- Les membres du jury du meilleur papier pour tout leur travail de relecture et de conseil,
- Les équipes du laboratoire A2SI de l'ESIEE et de l'Institut Gaspard Monge de l'Université Paris Est Marne-la-Vallée qui ont su répondre présent dès que nous avions besoin d'eux,
- Les étudiants de master 2 recherche de l'Institut Gaspard Monge et de l'école d'ingénieur IMAC pour leur travail de fourmis à nos côtés et leur bonne humeur.

Enfin je souhaite remercier les financeurs de ces journées et tout particulièrement la direction de l'ESIEE pour la mise à disposition de la salle Marcel Dassault, l'université Paris Est Marne-la-Vallée et la région Île-de-France pour leur soutien financier. Bien évidement, je remercie les partenaires financiers traditionnels des journées : l'AFIG et le GDR IG.

Sommaire

Détection de pupille par combinaison des critères morphologiques et colorimétriques.

Benjamin Raynal,Venceslas Biri

raynalb@esiee.fr,biri@univ-mlv.fr
Université Paris-Est
Laboratoire d'Informatique de l'Institut Gaspard Monge
UMR CNRS 8049
5 bd Descartes, 77454 Marne la Vallée Cedex 2, France

Abstract

Dans cet article nous étudions le problème de la détection de pupilles dans une image de résolution moyenne. On considérera que la connaissance de la position globale de l'oeil est déjà acquise, pour se concentrer sur le problème de la détection de la pupille dans une zone restreinte. Nous proposerons pour cela une méthode combinant des critères morphologiques et colorimétriques. Cette combinaison permettra, comme nous le démontrerons, une grande robustesse de détection. Cet algorithme sera également opitmisé, afin d'être utilisable en temps réel.

We study in this article the problem of pupil detection in a medium resolution image. The global location of the eye is supposed to be known and we focus on the pupil detection in a restricted area of the initial image. Our new method combine morphologic and colorimetric criteria to obtain a very robust detection of the eye center.

1. Introduction

Le suivi du regard est une discipline de vision par ordinateur, consistant à étudier la position des yeux de l'utilisateur, et à partir de là, à déterminer la direction fixée par celui-ci. Les applications du suivi du regard [Duc02] sont diverses, par exemple :

Interface homme-machine La principale application du suivi du regard est son utilisation en tant que système de pointage, dans le cas où l'utilisateur est dans l'incapacité d'utiliser ses bras, du fait de sa situation, ou de sa condition.

Etude de conception On peut aussi utiliser le suivi du regard sur des utilisateurs témoins, devant une affiche où un site internet, pour analyser quelles parties sont étudiées ou ignorées, afin d'optimiser leur impact.

Compression vidéo Il est également envisageable d'utiliser ce type de méthodes pour déterminer les régions d'intérêt dans une vidéo, afin de régler la qualité de compression en conséquence.

Etudes cognitives Une tout autre utilisation du suivi du regard consiste à l'utiliser pour étudier la façon dont on perçoit une scène, un texte, et comment on les analyse.

Les algorithmes de suivi du regard comportent généralement une phase de détection de l'oeil, de façon globale, suivi d'une phase de détection du centre de l'iris et de la pupille. Cette dernière se doit d'être robuste et précise, car d'elle dépend, bien entendu, la précision du suivi du regard.

Les principaux problèmes liés aux algorithmes de détection de pupille existants sont :

- le coût financier, certains nécessitant d'utiliser un type de caméra particulier (caméras infrarouges, par exemple), d'une valeur souvent trés élevée (entre 4000 et 50000 euros).
- la sensibilité au changement d'utilisateur ou de luminosité (cas notamment des méthodes utilisant un réseau de neurones [ZY02]).
- la durée et la difficulté de calibration de l'algorithme .

Ces problèmes rendent difficile d'accès le suivi du regard aux personnes en ayant le plus besoin : les personnes lourdement handicapées.

De plus, un algorithme de détection de pupille doit valider les contraintes suivantes :

Rapidité Permettre l'utilisation dans une application temps réel.
Précision Détecter le plus finement possible la pupille.
Robustesse Insensible au bruit et à l'environnement.
Efficacité Ne donner de résultat que si celui-ci est pertinent.

Nous présentons dans cet article une méthode de détection de pupille utilisant en entrée une image de résolution moyenne de l'oeil, et répondant aux critères ci-dessus. Cette méthode, de moindre coût car ne nécessitant aucun matériel hautement spécifique, sera conçue pour être temps réel. De plus, elle sera robuste pour tout utilisateur et dans un grand nombre d'environnements, d'illumination différente.

2. État de l'art

Si les techniques pour détecter la position globale de l'oeil sont nombreuses (détection des clignements de l'oeil [KT02], utilisation de caméra IR [JZ02, ZJ05b, ZJ05a], réseaux de neurones [SW95, VRR98, BMM00], données anthropomorphiques [SYW96], détection de la couleur de la peau [SW95, VRR98], utilisation de marqueurs [KR99]), celles pour détecter le centre de la pupille le sont moins.

On peut classifier ces méthodes en deux familles :

1. celles utilisant un apprentissage, basées sur l'apparence.
2. celles basées sur un modèle.

La première famille contient toutes les méthodes utilisant un réseau de neurones [SW95, VRR98], un modèle de Markov caché, ou toute méthode utilsant une base de données préalablement traitée. Du fait justement qu'elles nécessitent un apprentissage, elles sont sensibles aux changements d'utilisateur, voire même au changement de luminosité. De plus, cela implique une durée d'apprentissage, souvent non négligeable, voire une certaine difficulté à effectuer celui-ci pour un utilisateur quelconque. Ceci rend donc les algorithmes de cette famille particulièrement inadaptés à l'utilisation dans un suivi du regard utilisable par des personnes lourdement handicapées.

La seconde famille contient toutes les méthodes utilisant les propriétés de la pupille pour détecter celle-ci :

forme en partant de la carte des contours de l'image, on essaye alors de détecter la forme ressemblant le plus à un cercle, en utilisant par exemple la transformée circulaire de Hough [KBS75].
teinte En utilisant la couleur de la pupille, on cherche à détecter les composantes connexes de cette couleur répondant à divers critères géométriques [MMM06, KR99].

Un sous ensemble de cette seconde famille contient les méthodes utilisant un premier résultat approximatif, comme les méthodes infrarouges, où la pupille est détectée grossièrement grace au reflet de la lampe IR, où les méthodes se basant sur les résultats précédants, en utilisant par exemple le filtre de Kalman [ZJFL02]. A partir de ce premier résultat, il est plus aisé de détecter quels points appartiennent au contour de la pupille, et alors de calculer l'ellipse passant au mieux par ces points, au moyen de la méthode du moindre carré [ZY02].

Notre méthode sera quant à elle basée sur un modèle, et n'utilisant aucun résultat préalable, autre que la détection globale de la zone de l'oeil.

3. Architecture

Notre méthode peut se décomposer en deux passes successives :

- Une première passe permettra, dans toute l'image, de trouver la position approximative de la pupille.
- Une seconde détection permettra de localiser plus précisement le centre de la pupille, en utilisant le résultat de la première passe.

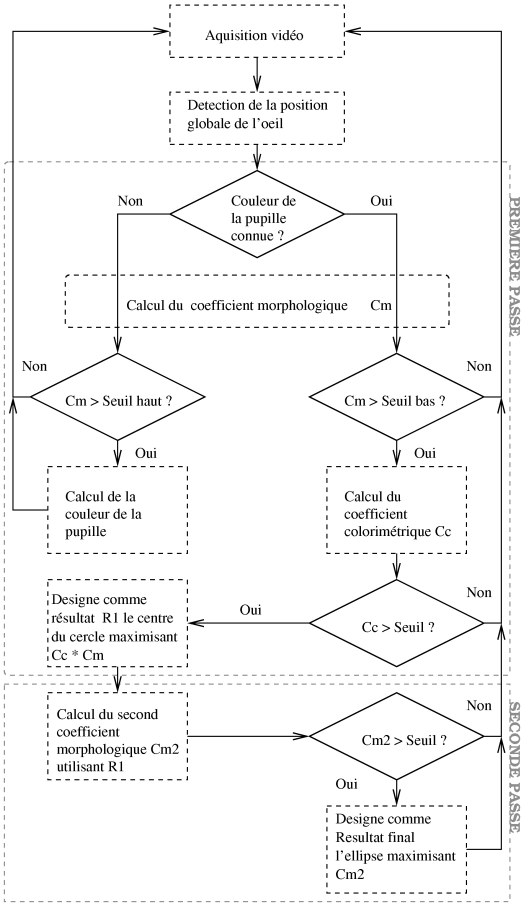


Figure 1: *Architecture globale de notre algorithme.*

Lors de la première passe, on calculera pour les candidats deux coefficients : un coefficient morphologique, basé

sur la forme circulaire de la pupille, et un coefficient colorimétrique, basé sur la couleur de la pupille. Pour qu'un candidat soit sélectionné, il doit avoir chacun de ses coefficients au dessus de leur seuil respectif. Si aucun candidat n'est sélectionné, l'algorithme ne retourne pas de résultat. Dans le cas contraire, il utilise comme premier résultat le candidat dont le produit des deux coefficients est maximal.

Durant la seconde passe, on calculera un second coefficient morphologique plus précis, en utilisant le résultat de la première passe.

Une première phase consistera à trouver la couleur de la pupille, qui servira par la suite à calculer le coefficient colorimétrique. Pour cela, on commence par une phase de détection des cercles ayant un coefficent morphologique au dessus d'un seuil haut, afin d'être sur de ne pas avoir une détection erronée. Le premier cercle au dessus de ce coefficient sera utilisé pour calculer la couleur de la pupille.

4. Coefficient morphologique

Le coefficient morphologique doit permettre de mesurer la ressemblance du contour de la pupille à un cercle où à une ellipse. Il est calculé sur une image binaire des contours C, produite par l'application du filtre de Canny [Can86]. Ce filtre à plusieures propriétés appréciables :

bonne détection faible taux d'erreur dans la signalisation des contours.

bonne localisation minimisation des distances entre les contours détectés et les contours réels.

clarté de la réponse une seule réponse par contour, insensible au bruit.

Cet opérateur est donc bien adapté à nos besoins, la propriété de bonne localisation permettant de ne pas perdre de précision, et la clarté de la réponse permettant de traiter le problème du bruit.

4.1. Choix entre ellipse et cercle

La pupille n'est circulaire dans l'image que lorsque l'oeil est orienté vers la caméra. Lorsqu'il est orienté dans d'autres directions, la forme de la pupille dans l'image s'approche plus d'une ellipse. Cependant, alors qu'un cercle n'est caractérisé que par trois paramêtres (ses coordonnées dans le plan, et son rayon), une ellipse en utilise cinq (ses oordonnées dans le plan, ses deux rayons axiaux, son angle de rotation).
Sachant que la complexité de l'algorithme dépendra du nombre de ces paramètres, nous avons porté notre choix sur la détection de cercle, afin d'avoir un temps de calcul le plus restreint possible. Une détection d'ellipse sera enviseageable lorsque les variations des paramètres auront été fortement restreintes.

4.2. Méthode de calcul

Pour calculer le coefficient morphologique d'un cercle de centre (x_c, y_c) et de rayon r_0, on va appliquer le masque adapté M sur l'image binaire des contours C (dont les valeurs sont égales à 1 en cas de présence d'un contour et 0 dans le cas contraire) grâce à la formule suivante :

$$k_m(x_c, y_c, r_0) = \frac{\sum_{m \in M} C(x_c + x_m, y_c + y_m) * M(x_m, y_m)}{\sum_{m \in M} M(x_m, y_m)}$$

Les masques sont bien entendus centrés en $(0,0)$.

4.3. Conception du masque

Une première idée consiste à utiliser un simple masque représentant le cercle de rayon R_0 désiré, de façon discrète. Cependant, du fait de l'imperfection de la circularité de la pupille dans l'image, peu de points du contour correspondent à un masque donné.
En nous inspirant des masques proposés par Tian et al. [TKC99], nous avons utilisé des masques d'anneaux, d'épaisseur proportionnelle fixe E. Pour tout cercle de rayon r_0, correspond un anneau de rayon interne $r_1 = r_0 - \frac{E}{2}$ et de rayon externe $r_2 = r_0 + \frac{E}{2}$. Cela permet certes de capturer l'intégralité du contour visible de la pupille, mais entraine évidemment une erreur de précision sur le centre de la pupille.
Pour limiter la marge d'erreur, nous avons décidé de pondérer les valeurs du masque, en utilisant une fonction gaussienne f sur la distance au cercle :

$$f(x,y) = \frac{1}{\pi E} e^{-\frac{\left(r_0 - \sqrt{x^2+y^2}\right)^2}{E}}$$

Ceci nous permet d'avoir tous les points du contours de la pupille masqués, sans trop toutefois sacrifier la précision.

4.4. Optimisation du calcul

Pour calculer l'ensemble des coefficients morphologiques, en chaque point de l'image I, et pour un ensemble de rayons entiers R, l'utilisation de convolutions se traduit par une complexité en $o(I_w * I_h * card(M_R))$, où $card(M_R)$ représente le nombre de points total pour l'ensemble des masques de rayon appartenant à R (voir l'algorithme 1).

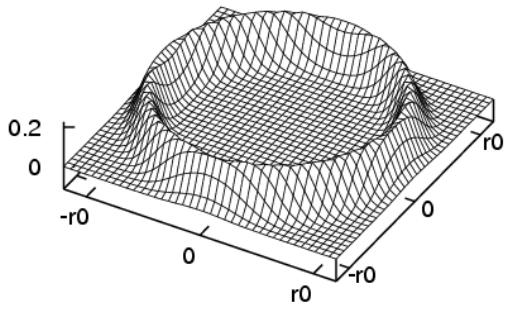


Figure 2: *Valeurs de la fonction gausienne f*

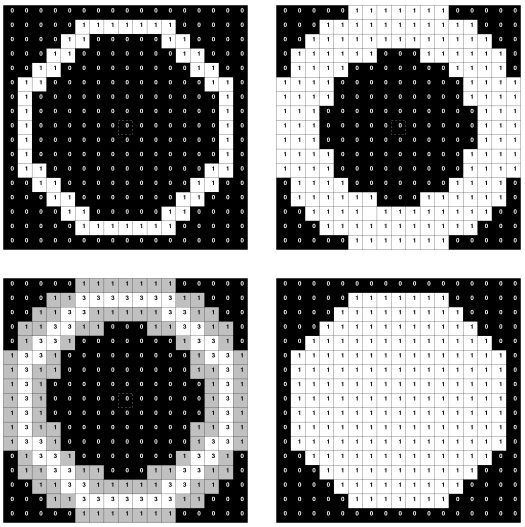


Figure 3: *Les différents types de masques correspondant à un même cercle : en haut, à gauche, masque de cercle, à droite, masque d'anneau. En bas, à gauche, masque d'anneau pondéré, à droite, masque de disque.*

Algorithm 1: Algorithme par convolution

```
Data: image I, double seuil, image M
Result: set<point> res
begin
    set<point> res ← ∅;
    foreach point p ∈ I do
        double km ← 0;
        double d ← 0;
        foreach point m ∈ M do
            km ← km + I(p+m) * M(m);
            d ← d + M(m);
        end
        km ← km/d;
        if km > seuil then
            res ← res ∩ {p};
        end
    end
end
```

En constatant que les pixels de valeur non nulle dans la carte des contours ne représentent qu'une petite part de la totalité de l'image, on peut utiliser la symétrie des masques pour réduire la complexité à $o(card(C) * card(M_R))$, où $card(C)$ est le nombre de points de valeurs non nulles dans l'image. On utilise pour cela un accumulateur A, c'est à dire un tableau de mêmes dimensions que l'image, où seront stockées les valeurs de coefficient morphologique de chaque point (voir l'algorithme 2).

Algorithm 2: Algorithme avec accumulateur

```
Data: image I, double seuil, set<point> C, image M
Result: set<point> res
begin
    set<point> res ← ∅;
    image A;
    double size ← 0;
    foreach point m ∈ M do
        size ← size + M(m);
    end
    seuil ← seuil * size;
    foreach point c ∈ C do
        foreach point m ∈ M do
            A(c+m) ← A(c+m) + M(m);
            if A(c+m) > seuil then
                res ← res ∩ {(c+m)};
            end
        end
    end
end
```

L'utilisation de l'algorithme 2 plutot que l'algorithme 1 permet de diviser la complexité du calcul du coefficient morphologique par $\frac{I_w * I_h}{card(C)}$, avec $card(C) \ll I_w * I_h$.

5. Coefficient colorimétrique

Le coefficient colorimétrique est utile en complément du coefficient morphologique, et permet de détecter des pupilles dans des yeux moins ouverts. En effet, l'ajout d'une contrainte sur un critère autre que la forme permet d'être moins restrictif sur celle-ci, sans pour autant sacrifier à la robustesse.
Ce coefficient sera calculé à partir de la couleur de référence de la pupille, et de la couleur moyenne à l'intérieur du cercle dont on veut le coefficient.
Dans toute cette section, les couleurs seront des vecteurs de trois dimensions (R,V,B), de valeur comprises dans l'intervalle $[0,1]$.

5.1. Calcul de la couleur moyenne dans un cercle

Pour calculer la couleur moyenne dans un cercle de centre (x_c, y_c) et de rayon r_0, on va utiliser le masque M_D du disque de rayon r_0 discrétisé, et l'appliquer à l'image couleur I de

la façon suivante :

$$\bar{V}(x_c, y_c, r_0) = \frac{\sum_{m \in M_D} I(x_c + x_m, y_c + y_m) * M_D(x_m, y_m)}{\sum_{m \in M_D} M_D(x_m, y_m)}$$

5.2. Acquisition de la couleur de la pupille

Pour pouvoir calculer ce coefficient, il nous faut d'abord acquérir la couleur de la pupille. Pour cela, on va détecter un cercle ayant une forte probabilité de correspondre à la pupille, en ne selectionnant les candidats qu'ayant un coefficient morphologique trés élevé. Lorsqu'un tel cercle est trouvé, on peut calculer la couleur de l'iris V_I en utilisant la formule de calcul de la couleur moyenne défini plus haut.

5.3. Méthode de calcul

Pour obtenir le coefficient colorimétrique d'un cercle C, on calcule la couleur moyenne $\bar{V}_C$ de son intérieur, puis on utilise la formule suivante :

$$k_c(x_c, y_c, r_0) = 1 - \frac{\|\bar{V}_C - V_I\|_2}{\sqrt{3}}$$

6. Raffinement du résultat

Pour gagner en précision, on utilise ce premier résultat comme base de départ pour une détection plus fine : nous allons maintenant chercher à détecter des ellipses, de centre proche de celui du premier résultat, et de rayons avoisinant celui du premier résultat.
Afin de supprimer le plus de contours inutiles (n'appartenant pas au contours de la pupille), nous avons développer un gradiant radial, dont la direction est orientée à partir d'un centre donné, ici, le centre du premier résultat. C'est sur la carte des contours donnée par ce gradiant que nous appliquons la détection d'ellipses.

6.1. Gradient radial

Nous avons cherché à concevoir le gradient le plus adapté à nos besoins, ayant les mêmes propriétés que le filtre de Canny, mais optimisant les contours de toute forme convexe entourant un point donné. En nous inspirant des travaux de Li et al. [LP06], nous avons décidé d'utiliser un gradient défini en fonction d'un centre C donné.

Soit I l'image de base, G_C est l'image qui en chaque point P contient l'approximation de la dérivée du point correspondant, selon l'angle θ que fait $\vec{CP}$ avec le vecteur $(1,0)$. G_C est obtenue par convolution de I par N_θ, N_θ étant un noyau de convolution défini en fonction de θ (voir la figure 4).

Une fois G_C obtenu, on en extrait les maxima locaux

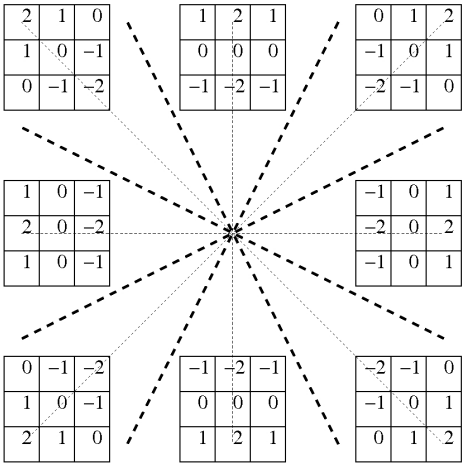


Figure 4: *Les différents noyaux de convolution selon la position par rapport au centre.*

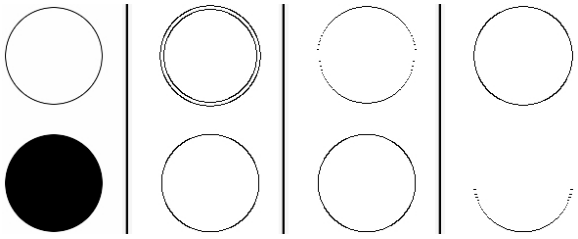


Figure 5: *De gauche à droite : image de base, image traitée avec Canny, image traitée avec notre opérateur centré sur le disque du bas, image traitée avec notre opérateur centré sur le cercle du haut.*

selon la direction du vecteur $\vec{CP}$. On seuille ensuite le résultat par hysteresis.

Les propriétés de notre filtre sont les suivantes :

- marque bien, et **uniquement** les contours d'une zone foncée vers une zone plus claire, en partant du centre, perpendiculaires à la direction au centre.
- une bonne localisation (meilleure que Canny, dans certains cas).
- une réponse minimale.

La première propriété peut paraitre étrange, mais est utile pour pouvoir détecter plus efficacement certains yeux clairs qui ont un bord de pupille mince plus sombre (voir la figure 5). Alors que le filtre de Canny donnera dans ce cas deux contours trés proches, notre opérateur n'en donnera qu'un. Pour détecter également les contours d'une zone claire vers une zone foncée, il suffirait d'utiliser les valeurs absolues de G_C.

Ce filtre a pour autre propriété intéressante de trés bien

détecter les contours perpendiculaires aux rayons partant de son centre, et d'ignorer ceux qui y sont parallèles. Il en découle une trés bonne détection des contours d'un cercle contenant le centre du filtre, et une bien moins bonne pour ceux ne le contenant pas (voir les dernières colonnes de la figure 5). C'est donc une sécurité supplémentaire qui permet de renforcer la contrainte de robustesse.

6.2. Détection d'ellipse dans la zone restreinte

Une fois notre nouvelle carte des contours calculée, on l'utilise comme base pour détecter des ellipses avec de fortes contraintes, décrites ci-dessous. Soit r_0 le rayon du premier résultat, C_0 son centre,et ε une constante fixée. Définissons la fenêtre W de coté r_0, et de centre C_0. Les ellipses détectées sont celles dont le centre se situe dans W, et dont les rayons axiaux sont compris dans $[r_0 * (1-\varepsilon), r_0 * (1+\varepsilon)]$.

Les masques utilisés pour détecter ces ellipses sont des masques d'anneaux ellipsoïdaux pondérés, selon une adaptation de la formule utilisée pour pondérer nos masques d'anneaux : pour une ellipse d'axe semi mineur r_x parallèle à l'axe des abscisses et d'axe semi majeur r_y, on défini

$$\alpha_{x,y} = \sqrt{\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2}}$$

le coefficient de mise à l'echelle de l'ellipse pour qu'elle passe par le point (x,y). On peut alors définir la fonction de pondération en fonction de la distance à l'ellipse comme ci-dessous :

$$g(x,y) = \frac{1}{\pi E} e^{\frac{-\left(\left|\frac{\alpha_{x,y}-1}{\alpha_{x,y}}\right|\sqrt{x^2+y^2}\right)^2}{E}}$$

7. Résultats expérimentaux

Dans le but de mesurer la précision et l'efficacité de notre algorithme, nous l'avons testé sur 6 séries d'images d'oeil, provenant de différents utilisateurs aux couleurs de prunelles/d'iris différentes. Chaque série contient 51 images de résolution moyenne de 150*150 pixels et représentant la zone globale de l'oeil.

A partir de ces données, nous avons compté le nombre de résultats obtenu pour notre algorithme, avant et après la seconde passe, et le meilleur résultat de la transformée circulaire de Hough, pour avoir une référence. Nous avons également mesuré la précision des algorithmes, en utilisant la distance entre le centre trouvé par l'algorithme et celui du cercle désigné manuellement, servant de référence. Cette distance est ensuite divisée par le rayon du cercle de référence, afin d'avoir une mesure relative. Enfin, pour évaluer l'efficacité, nous utilisons la proportion de résultats valides (à une distance raisonable du centre de référence), par rapport au nombre de résultats total.

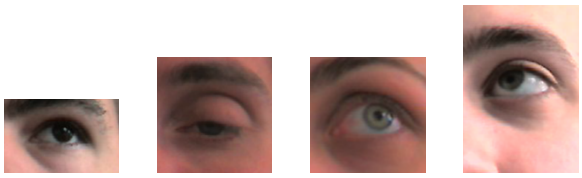


Figure 6: *Echantillon d'images de test.*

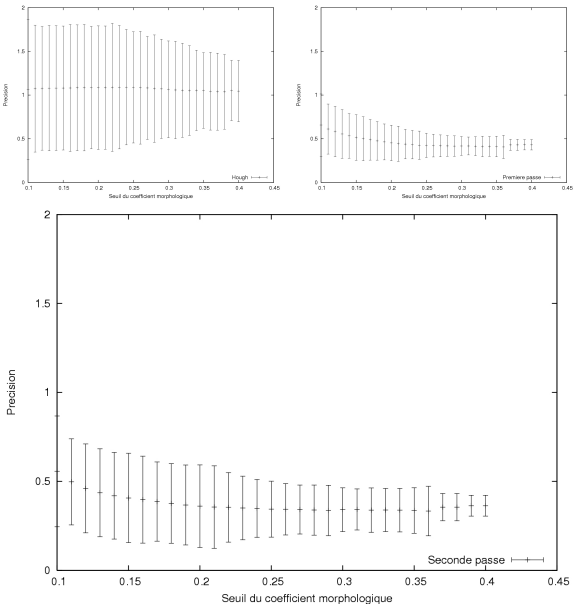


Figure 7: *Selon le seuil de coefficient morphologique. En haut à gauche, la transformée de Hough, à droite, la première passe de notre algorithme. En bas, seconde passe de notre algorithme.*

On constate que le nombre de résultats obtenus diminue en même temps que la précision augmente, ce qui est normal, étant donné que l'on supprime les résultats les moins probants pour gagner en précision.
On observe que la seconde passe apporte un gain de précision non négligeable, mais entraine également une baisse du nombre de résultats. Cependant, pour un seuil du coefficient morphologique aux alentours de 0.2, on obtient un bon gain de précision, sans trop perdre de résultats. (voir la figure 7).
Il faut aussi prendre en compte que les résultats sont ici des moyennes, une adaptation plus fine des seuils en fonction de l'utilisateur donnera de meilleurs résultats.

Pour ce qui est de la vitesse d'exécution, notre algorithme tourne sur la machine de test (Pentium 4 2.8Ghz) à 50 Hz environ.

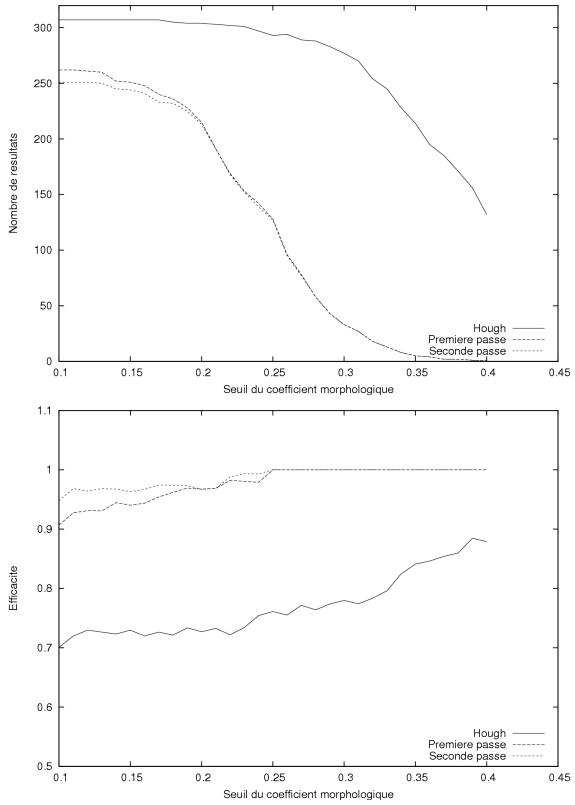


Figure 8: *Nombre de résultats et efficacité selon le seuil de coefficient morphologique (Seuil du coefficient colorimétrique à 0.85).*

8. Conclusion

Nous avons développé un algorithme de détection de pupilles répondant aux critères fixés :

précision étant donnée la qualité de l'image, et les besoins matériels, notre algorithme à une précision tout à fait acceptable, grace à la seconde passe.

efficacité L'utilisation de seuils nous assure que le résultat obtenu est de même forme et de même couleur que la pupille. Du fait de la localité de l'image, la probabilité que ce résultat corresponde à la pupille est trés élevé.

robustesse du fait de l'utilisation du filtre de Canny pour produire les données de départ, notre algorithme est insensible au bruit, ainsi qu'à la couleur des yeux, grace à notre méthode d'acquisition de la couleur de la pupille.

rapidité avec une fréquence de 50 Hz en moyenne sur une machine de milieu de gamme, notre algorithme peut être utilisé pour une application temps réel.

Cet algorithme permet donc de faire du suivi du regard en temps réel, tout en ne nécessitant qu'une webcam de bonne qualité, pour un prix avoisinant les 100 euros.
Enfin, la calibration de l'algorithme ne nécessite que peu de réglages, dont beaucoup peuvent être automatisés en utilisant des données fournies par d'autres parties du logiciel de suivi du regard. Tout ceci permet l'utilisation de cet algorithme dans un logiciel accessible aux personnes lourdement handicapées, notre objectif est donc atteint.
Pour améliorer les résultats, diverses pistes sont envisageables : utiliser le filtre de Kalman [Kal60], afin de s'aider des résultats précédents pour déterminer le centre de la pupille et ainsi gagner en efficacité, où encore utiliser des techniques sous pixels [ZY02], afin de gagner en précision.

References

[BMM00] BETKE M., MULLALLY W., MAGEE J.: Active detection of eye scleras in real time. In *Proceedings of the IEEE Workshop on Human Modeling, Analysis and Synthesis, June 2000.* (2000).

[Can86] CANNY J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 6 (1986), 679–698.

[Duc02] DUCHOWSKI A. T.: A breadth-first survey of eye-tracking applications. *Behav Res Methods Instrum Comput 34*, 4 (November 2002), 455–470.

[JZ02] JI Q., ZHU Z.: Eye and gaze tracking for interactive graphic display. In *Int. Symp. on Smart Graphics* (2002).

[Kal60] KALMAN R.: A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82*, Series D (1960), 35–45.

[KBS75] KIMME C., BALLARD D., SKLANSKY J.: Finding circles by an array of accumulators. *Commun. ACM 18*, 2 (1975), 120–122.

[KR99] KIM K., RAMAKRISHNA R.: Vision-based eye-gaze tracking for human computer interface. In *IEEE International Conf. on Systems, Man, and Cybernetics, Tokyo, Japan* (1999).

[KT02] KAWATO S., TETSUTANI N.: Detection and tracking of eyes for gaze-camera control. In *VI '02 : Vision Interface 2002* (2002).

[LP06] LI D., PARKHURST D.: Open-source software for real-time visible-spectrum eye tracking. In *COGAIN Conference* (2006).

[MMM06] MERAD D., METZ S., MIGUET S.: Eye and gaze tracking algorithm for collaborative learning system. In *International Conference on Informatics in Control, Automation and Robotics (IFAC/ICINCO 2006)* (2006).

[SW95] SCHIELE B., WAIBEL A.: Gaze tracking based on face-color. In *International Workshop on Automatic Face and Gesture Recognition, Zurich* (1995), pp. 344–349.

[SYW96] STIEFELHAGEN R., YANG J., WAIBEL A.: A

model-based gaze tracking system. In *IEEE International Joint Symposia on Intelligence and Systems* (1996), pp. 304–310.

[TKC99] TIAN Y., KANADE T., COHN J.: Dual-state parametric eye tracking. In *International Conference on Face and Gesture Recognition, 1999.* (1999).

[VRR98] VARCHMIN A., RAE R., RITTER H.: Image based recognition of gaze direction using adaptive methods. *Lecture Notes in Computer Science 1371* (1998), 245+.

[ZJ05a] ZHU Z., JI Q.: Eye gaze tracking under natural head movements. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (2005), vol. 1, pp. 918–923.

[ZJ05b] ZHU Z., JI Q.: Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. *Comput. Vis. Image Underst. 98*, 1 (2005), 124–154.

[ZJFL02] ZHU Z., JI Q., FUJIMURA K., LEE K.: Combining kalman filtering and mean shift for real time eye tracking under active ir illumination. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 4* (2002).

[ZY02] ZHU J., YANG J.: Subpixel eye gaze tracking. In *FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition* (2002).

Indices de formes : de la 2D vers la 3D Application au classement de noyaux de cellules

Guillaume THIBAULT, Caroline DEVIC, Bernard FERTIL, Jean-Luc MARI et Jean SEQUEIRA

Université Aix-Marseille II
Laboratoire LSIS - UMR CNRS 6168
Equipe I&M (Images et Modèles)
ESIL - Campus de Luminy case 925
163 Avenue de Luminy 13288 Marseille

Abstract

In this paper, we present a study on the characterization and the classification of binary digital shapes. This study is performed using a set of values obtained by the computation of "shape indexes". To get these indexes, we extract a set of data called "measures" from the 2D shape, like for example a shape's surface and perimeter. Then these measures are used as parameters of a function bringing a real value which gives information about geometrical and morphological features on the shape to analyze. A model of shape and texture characterizations is built based on this numerical information. It is used to classify cells nuclei in order to diagnose patients affected by the Progeria disease.

***Résumé** : Dans cet article, il est présenté une étude sur la caractérisation et la classification de formes numériques binaires, effectuée à l'aide d'un ensemble de valeurs obtenues par le calcul d'indices de formes. Pour calculer ces indices, il est nécessaire d'extraire de la forme un ensemble de données appelées "mesures", par exemple l'aire et le périmètre de la forme. Ces mesures sont ensuite utilisées comme paramètres d'une fonction apportant une valeur réelle ou discrète qui nous renseigne sur les caractéristiques géométriques et morphologiques de la forme ou du volume à analyser. Toutes ces informations numériques permettent de construire un modèle qui décrit la forme et qui peut être ensuite utilisé par exemple pour classifier des formes et une application est proposée pour diagnostiquer des noyaux de cellules chez des patients atteints par la maladie de la Progéria.*

1. Introduction

La reconnaissance de formes est une partie majeure de l'intelligence artificielle qui vise à automatiser le discernement de situations typiques au niveau de la perception. Elle est un enjeu majeur pour de très nombreuses applications : la reconnaissance des caractères manuscrits (numérisation des livres, lecture automatique des lettres postales et des chèques bancaires, etc.), la vidéo surveillance (reconnaissance faciale), l'imagerie médicale (échographie, scanner, imagerie par résonance magnétique), la télédétection, etc.

Au cœur de la reconnaissance de formes, il y a une première étape incontournable : la caractérisation de formes. En effet, afin de pouvoir reconnaître un objet ou un individu, il faut tout d'abord le décrire et donc définir ses caractéristiques (morphologiques, géométriques, texturelles, etc.) et ensuite retrouver et identifier ces mêmes caractéristiques sur la source numérique à analyser. Pour cela il est souvent nécessaire d'étudier les objets selon deux critères : le premier est la forme avec des méthodes d'analyse globale (par exemple signature polaire [THK06], histogrammes de projections [SR04], etc.) ou de contours (par exemple chaînes de Freeman [IPSV97], MSGPR [KR06], etc.) et le deuxième la texture [CDT95, Cos01, Mav01].

Pour réaliser l'analyse de la forme, la caractérisation à l'aide d'indices de formes est de plus en plus utilisée [AAB07, Ben02, IP97, TLG*03, URW*02, ZR04] et notamment en classification avec l'utilisation fréquente des méthodes par apprentissage [AGW97, SEB*03]. La souplesse d'utilisation, la simplicité de mise en oeuvre et la facilité d'utilisation avec un classificateur font de cette méthode un

choix pertinent pour de nombreux problèmes.
Le but de cet article est de créer un modèle afin de classer des noyaux de cellules de patients atteints par la maladie de la Progéria (également connue sous le nom de syndrome de Hutchinson-Gilford). Cette maladie orpheline (une centaine de cas dans le monde) de type laminopathie [SGBC*03] provoque un vieillissement accéléré du patient. Nous disposons d'un ensemble de noyaux de cellules prélevés chez les patients, ainsi que d'une expertise de ces noyaux. Cette expertise révèle que la forme du noyau (normale ou bien *boursouflée*) joue un rôle principal dans le diagnostic, mais des informations complémentaires sont obtenues par une analyse de la texture concernant l'homogénéité du noyau. Pour observer la texture, les images de noyaux sont acquises à l'aide d'un microscope à fluorescence et on utilise un marqueur de type FITC afin de voir la répartition des lamines AC.
La première partie de ce document définit les notions de mesure, d'indice de forme et leurs principales propriétés (section 2) ainsi que la façon d'utiliser les indices pour la reconnaissance de formes (section 2.1). Ensuite, le modèle mis en œuvre pour résoudre le problème est présenté (sections 3 et 4), étudié et validé.

2. Indices de formes et mesures

Les indices de formes ont été présentés pour la première fois en 1976 dans le livre de Santalo [San76] relatif aux propriétés mathématiques des formes convexes. On trouve la définition et les propriétés des indices de formes dans [CC85, Fil95].

Définition 2.1 (Indice de formes) *On appelle indice de formes tout paramètre, cœfficient ou combinaison de cœfficients permettant de donner des renseignements chiffrés sur la forme de l'objet.*

De plus, les indices doivent avoir les propriétés suivantes :

Propriété 2.2 (Indice de formes 1)

1. *Etre sans dimension.*
2. *Etre invariants par homothétie.*
3. *Etre invariants par rotation et translation.*
4. *S'appliquer à des formes connexes simples donc homéomorphes au disque.*

Le calcul d'un indice de formes est équivalent au calcul de la valeur d'une fonction à plusieurs variables. On aimerait alors que cette fonction soit bijective, mais deux contre-exemples sont donnés par les indices de concavité qui valent 1 pour toutes les formes convexes et l'indice d'allongement par le diamètre (défini plus loin [CC85]) pour le disque et le carré :

$$Allongement(Disque) = Allongement(Carre) = \frac{1}{2}$$

En revanche, la propriété suivante est vraie :

Propriété 2.3 (Indice de formes 2) *Soit F_1 une forme et I un indice de formes tel que $I(F_1) = \alpha$, avec $\alpha = cste \in \mathbb{R}$. Alors si pour toute forme F_2 quelconque, on a $I(F_2) \neq \alpha$ cela implique $F_1 \neq F_2$.*
Ce qui peut s'écrire :

$$Si\ \exists F_1,\ \exists F_2, \exists I\ /I(F_1) \neq I(F_2) \Rightarrow F_1 \neq F_2$$

Cette dernière propriété implique qu'un indice seul ne peut pas identifier une forme, mais qu'il "renseigne" sur la forme étudiée et sur les caractéristiques qu'elle ne possède pas.

La majorité des indices sont créés à partir d'une égalité ou d'une inégalité inhérente à la forme que l'on souhaite caractériser. Dans [Bon20] et [San76] les auteurs démontrent une série d'inégalités propres aux formes convexes dans un espace continu :

$$P^2 - 4\pi A \geqslant \frac{1}{2}\left[(P - 2\pi\rho_i)^2 + (2\pi\rho_e - P)^2\right]$$

$$P^2 - 4\pi A \geqslant \pi^2(\rho_e - \rho_i)^2 \text{ ou } P^2 - 4\pi A \geqslant \pi^2(\rho_{max} - \rho_{min})^2$$

avec A l'aire, P le périmètre, ρ_i (resp. ρ_e) rayon de la plus grande (resp. petite) boule inscrite (resp. circonscrite [Gär99]), ρ_{max} (resp. ρ_{min}) le rayon de courbure maximum (resp. minimum). Toutes ces inégalités font intervenir différents paramètres de la forme que l'on appelle "*mesures*" (figure 1). Le calcul de ces mesures est une étape incontournable pour pouvoir calculer des indices de formes, car tous les indices utilisent au moins une mesure.
La possibilité de construire un indice à partir d'une égalité (ou inégalité) propre à la forme que l'on souhaite caractériser est l'avantage majeur des indices de formes. Cela permet d'utiliser les indices pour tout type de forme. De plus, la complexité de calcul d'un indice réside dans la complexité d'extraction des mesures qui le compose. Ces avantages rendent cette technique particulièrement bien adaptée en classification.

Définition 2.4 (Mesure) *On appelle "mesure" d'une forme toute valeur ou ensemble de valeurs numériques mesurées sur la forme.*

Les mesures possèdent ou non des dimensions : trois dimensions (le volume), deux dimensions (la surface), une dimension (le périmètre, le diamètre euclidien, la longueur de l'axe principal, etc.) et aucune dimension (le nombre de composantes connexes, le nombre de trous, etc.). Les mesures sans dimension possèdent toutes les caractéristiques de la propriété 2.2 et sont par conséquent des indices de formes.

2.1. Indices de formes et classement

Le but de la classification est d'associer à chaque individu étudié une classe d'appartenance. Dans le problème présent l'objectif est de déterminer si un noyau de cellule est "*sain*" ou "*pathologique.*"
Les méthodes de classement se divisent en deux grandes

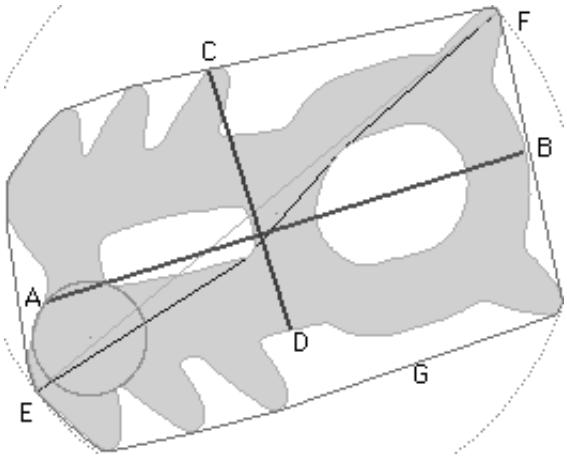


Figure 1: *Exemples de mesures : surface (gris clair), AB et CD axes principaux, G enveloppe convexe, EF diamètre géodésique (noir) et diamètre Euclidien (gris clair), plus petite boule circonscrite (pointillés), A plus grande boule inscrite.*

familles : les méthodes dites supervisées (appelées aussi méthodes de classement) et non supervisées. Les méthodes supervisées font intervenir une expertise et sont par conséquent généralement beaucoup plus puissantes. Dans notre cas, nous bénéficions de l'expertise des biologistes et généticiens qui ont déterminé les classes (sain et pathologique) et les sous classes (forme normale et forme boursouflée, texture homogène et texture non homogène, etc.), ce qui nous permet d'utiliser des méthodes supervisées.

Les méthodes de classement sont ce que l'on appelle des méthodes d'apprentissages. Leur but est de construire un modèle de classement en fonction des données à classer. Bien que s'appliquant à un problème spécifique, le modèle doit être capable de généraliser (au sens des données). Pour cela, les données sont séparées en deux groupes : un échantillon d'apprentissage et un échantillon de validation. Le classificateur doit garder les mêmes performances lors de l'apprentissage et de la validation.

Pour mettre en œuvre une méthode de classement, il faut au préalable construire un vecteur caractéristique de l'individu étudié. Le vecteur doit être pertinent au problème posé afin de permettre un bon classement et une bonne prédiction. Le risque majeur lorsque l'on fournit trop de caractéristiques au classificateur est l'apprentissage par *cœur*. Plus la dimension du vecteur est grande, plus le modèle sera adaptable et donc plus la classement sera bon, mais plus la validation du modèle à l'aide d'individus non utilisés dans la phase d'apprentissage sera mauvaise. Il faut alors systématiquement valider chaque modèle construit et obtenir le meilleur compromis entre bonne classification et bonne prédiction. Dans notre problème, le vecteur caractéristique sera composé des indices de formes afin de caractériser la forme et des caractéristiques Haralick [HSD73, Har79] pour la texture.

La méthode de classification choisie pour le modèle est la régression logistique [DS89]. C'est un modèle linéaire particulièrement adapté pour les problèmes de classement à deux classes. Elle réalise une analyse statistique des données de l'ensemble d'apprentissage et utilise une fonction de distribution logistique pour discriminer les données :

$$P = P(Y/x) = \frac{e^{f(x)}}{1+e^{f(x)}}$$

avec $x = (x_1, ...x_n)$ le vecteur caractéristique de la donnée en entrée, $f(x) = \sum_i \alpha_i x_i$ et $P(Y/x)$ la probabilité conditionnelle P de la variable x d'appartenir à la classe Y. Pour estimer les coefficients α_i du modèle, on utilise le plus souvent la méthode du maximum de vraisemblance qui maximise la probabilité d'obtenir les valeurs observées sur les échantillons de l'ensemble d'apprentissage. Elle consiste à rechercher les paramètres qui optimisent la fonction de vraisemblance $£(\alpha, Y) = P^Y[1-P]^{1-Y}$.

3. Caractérisation de la forme des noyaux de cellules à l'aide des indices de formes

Nous disposons d'un ensemble de plus de trois milles noyaux de cellules (figure 2) prélevés chez des patients atteints par la maladie de la Progéria. Ces noyaux ont tout d'abord été classés manuellement et il est apparu que le critère de forme est l'élément de diagnostic décisif dans plus de 90% des noyaux. Il est ainsi primordial de construire un modèle de classification qui caractérise la forme du noyau. Pour cela, nous disposons d'un ensemble de douze indices de formes existant dans la littérature scientifique, mais il apparu nécessaire de créer trois nouveaux indices pour répondre plus spécifiquement au problème posé. La liste complète des indices utilisés est en annexe de cet article.

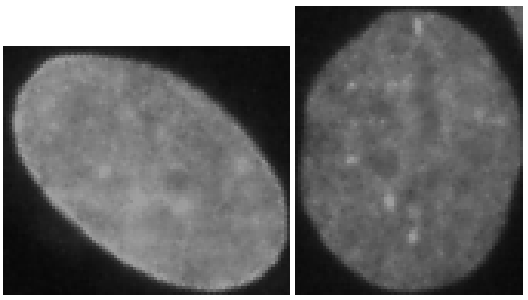


Figure 2: *Deux noyaux de cellules.*

3.1. Deux nouveaux indices de formes

Dans la section 2, nous avons vu que les indices de formes peuvent être créés à partir d'une égalité ou d'une inégalité

inhérente à la forme que l'on souhaite caractériser. Cette propriété est l'avantage majeur de cette technique, car elle rend l'utilisation des indices de formes applicable à tout type de forme.
Les noyaux de cellules étudiés ont une forme quasi elliptique lorsqu'ils sont sains. Il est donc utile de construire un indice qui permet de caractériser des ellipses. Il est basé sur la formule de l'aire d'une ellipse : $A = \pi ab$, avec a le demi grand axe et b le demi petit axe. Or dans le cas d'une ellipse, certaines mesures sont égales aux paramètres de l'aire : $R_{max} = a$ et $R_{min} = b$ ou encore $R_{max} = \frac{1}{2}L_{AP1}$ et $R_{min} = \frac{1}{2}L_{AP2}$, avec R_{max} (resp. R_{min}) le plus grand rayon (resp. plus petit) et L_{AP1} (resp. L_{AP2}) la longueur de l'axe principal (resp. la longueur de l'axe secondaire). Ces égalités permettent de construire deux indices de formes :

$$\psi_{ellipse} = \frac{\pi R_{min} R_{max}}{A} \; ou \; \frac{\pi}{4}\frac{L_{AP1}L_{AP2}}{A} \in [0,1]$$

Le dénominateur et le numérateur sont égaux dans le cas d'une ellipse et l'indice vaut 1.
Lorsque les noyaux ne sont pas sains, il est extrêmement fréquent qu'ils ne soient pas convexes et donc qu'ils possèdent des points de concavité. Pour compter ces points de concavité dans le cas des noyaux, il est possible de calculer le nombre de composantes connexes N_{Cce} issues de la soustraction de la forme à son enveloppe convexe. Cet indice sera utilisé sous sa forme normée $\psi_{N_{Cce}}$ dans le classificateur :

$$N_{Cce} = card(EnveloppeConvexe(F) \backslash F)$$

$$\psi_{N_{Cce}} = \frac{1}{1+N_{Cce}} \in]0,1]$$

Cet indice vaut 1 si aucune composante d'écart n'est trouvée et plus la forme a des points de concavité, plus l'indice tend vers 0.
Mais dans la pratique, on ne peut considérer comme composante connexe, des composantes dont la taille est de l'ordre du pixel et qui sont dues à des erreurs de discrétisation. De plus, dans les éléments de diagnostic des noyaux, la taille et le nombre des composantes connexes N_{Cce} doivent être pris en compte. Il faut trouver un ou plusieurs seuils de taille et de nombre à partir des éléments de diagnostic. Pour cela, nous avons réalisé une étude systématique du pourcentage de bonne classification en fonction de la taille et du nombre de composantes connexes d'écarts (figure 3).

Cette analyse met en exergue l'utilité de l'indice $\psi_{N_{Cce}}$ dans le cas de noyaux non convexes ayant soit un point de concavité d'au moins trente-deux pixels soit deux points de concavité d'au moins douze pixels (figure 4). Ces deux seuils sont utilisés de manière équivalente dans $\psi_{N_{Cce}}$ et leur combinaison permet ainsi d'obtenir un taux de bonne classification de plus de 90% par rapport au sous diagnostic de forme avec ce seul indice.

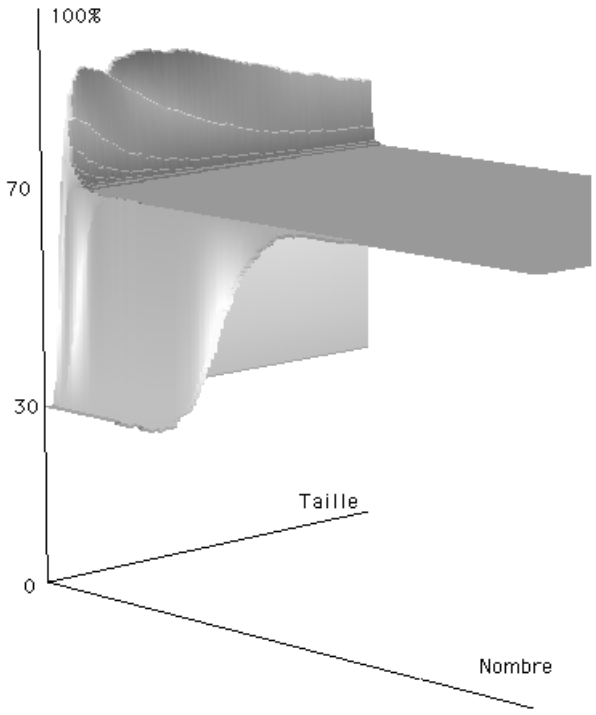


Figure 3: *Surface représentant le pourcentage de bonne classification des noyaux en fonction du nombre et de la taille des composantes connexes d'écarts.*

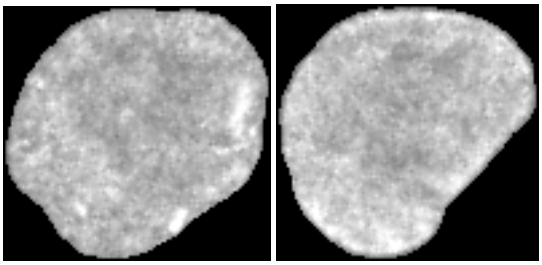


Figure 4: *Concavité : à gauche, un noyau avec deux points de concavité d'au moins douze pixels, à droite, un noyau avec un seul point de concavité d'au moins 32 pixels.*

3.2. Construction et validation du modèle de caractérisation de la forme des noyaux

Nous disposons maintenant d'un large éventail d'indices de formes afin de caractériser la forme des noyaux de cellules : les indices déjà existants dans la littérature et deux nouveaux indices créés pour répondre spécifiquement au problème traité. Il convient désormais de construire une combinaison linéaire des différents indices et de démontrer la pertinence du modèle créé pour répondre aux critères de formes.

Il faut choisir un nombre d'indices de formes pertinents afin d'obtenir un taux maximum de bonne classification et de validation, tout en conservant des propriétés de bonne généralisation (cf. section 2.1).
Pour cela, nous avons commencé par tester le modèle contenant tous les indices, puis au fur et à mesure les indices les moins pertinents ont été retirés. Pour la validation, chaque modèle a été construit à partir de 50% des noyaux choisis aléatoirement (résultats d'apprentissage, soit environ 1400 noyaux), puis validé sur les 50% restant (résultats de validation). Les figures 5 et 6 contiennent les différentes courbes construites pendant le test du modèle, sa validation ainsi que son efficacité.

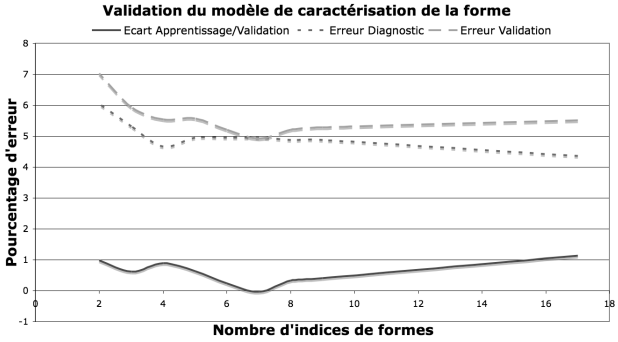


Figure 5: *Pourcentage de bonne classification lors de la validation du modèle de caractérisation de la forme. Le graphe montre l'intérêt d'utiliser les sept indices les plus pertinents.*

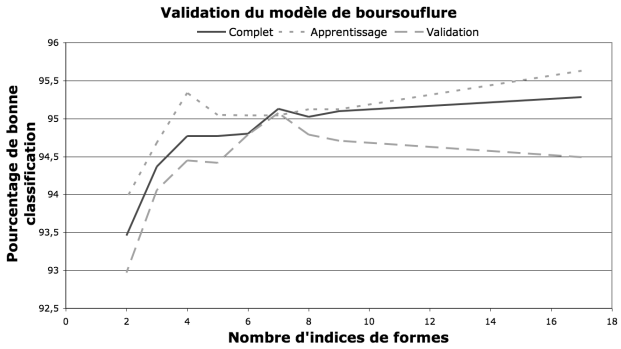


Figure 6: *Pourcentage d'erreur lors de la validation du modèle : les deux courbes pointillées convergent l'une vers l'autre pour sept indices, puis diverge si on augmente le nombre d'indice.*

Sur la figure 6 on peut constater que les courbes en pointillés représentant les erreurs de diagnostic et de validation convergent pour un nombre d'indices égal à sept. On voit également que l'écart entre ces courbes augmente de manière monotone au delà de sept indices. Ces deux graphiques démontrent la validité de notre modèle qui permet un taux de bonne classification de la forme supérieur à 95% pour sept indices.

4. Caractérisation de la texture

Bien qu'ayant obtenu de très bons résultats avec le modèle de forme, celui-ci ne répond qu'à 89% du problème final, qui est la classification des noyaux en "sain" ou "pathologique". Pour compléter le modèle, il faut pratiquer une analyse de la texture et plus particulièrement une analyse de l'homogénéité du noyau.
Afin d'observer les noyaux, les experts utilisent un marqueur fluorescent de type FITC dans le microscope. Ce marqueur réagit à la présence des lamines AC dans le noyau et révèle leur répartition qui doit être homogène au centre et légèrement plus importante sur le bord. Mais en pratique, il est quasiment impossible que le marqueur se répartisse régulièrement sur la totalité du noyau. Cette inégalité de répartition pourrait laisser croire que les lamines AC ne sont pas réparties de manière homogène. Forts de cette information, les experts considèrent comme texture non homogène, uniquement les noyaux ayant une texture "fortement non homogène" (figure 7). Pour éviter ce problème pendant l'analyse, nous appliquons au préalable un filtrage passe-bas de type Gaussien sur la totalité de la texture :

$$I(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$$

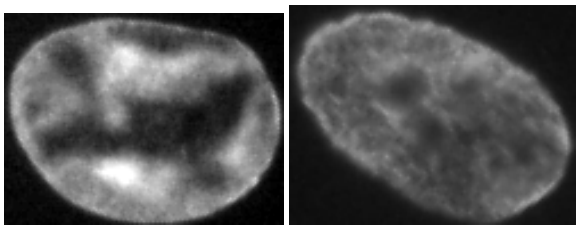


Figure 7: *Deux noyaux de cellules possédant une texture fortement non homogène.*

Pour caractériser la texture, nous construisons la matrice de co-occurence, puis nous calculons neuf caractéristiques Haralick [HSD73, Har79] :

1. La moyenne $m = \sum_x \sum_y p(x,y)$
2. L'écart type $\sum_x \sum_y (p(x,y) - m)^2$
3. Le contraste $\sum_{n=0..N_g} n^2 \{\sum_{i=1..N_g, |i-j|=n} \sum_{j=1..N_g} p(i,j)\}$
4. L'angle du moment du second ordre
5. La corrélation
6. L'entropie $\sum_x \sum_y p(x,y) log(p(x,y))$
7. L'homogénéité $\sum_x \sum_y \frac{1}{1+(x-y)^2} p(x,y)$
8. La dissimilarité $\sum_x \sum_y |x-y| p(x,y)$
9. L'inertie $\sum_x \sum_y (x-y)^2 p(x,y)$

avec $p(x,y)$ l'élément (x,y) de la matrice de co-occurence, N le nombre de pixels qui compose la texture à analyser, N_g le nombre de niveaux de gris. L'*angle du moment du second ordre* (que l'on appelle aussi *énergie*) mesure l'uniformité de la texture. Plus la texture est uniforme, moins il y a de transitions de niveaux de gris et donc plus la somme des carrés des éléments de la matrice est faible. L'homogénéité est d'autant plus élevée que l'on retrouve souvent le même couple de pixels, ce qui est le cas lorsque le niveau de gris est uniforme ou quand il y a périodicité spatiale. Les autres valeurs caractéristiques apportent d'autres informations sur la texture, moins intéressantes mais tout de même utilisées dans notre modèle.
Contrairement à l'expertise de la forme, l'expertise de la texture a fourni deux classes ("*homogène*" et "*non homogène*") totalement déséquilibrées ; il y a environ vingt fois plus d'individus dans la classe homogène. Pour construire efficacement le modèle, il faut équilibrer les classes pendant la phase d'apprentissage. Pour cela, l'apprentissage a été réalisé avec les noyaux dont la texture est "non homogène" et un nombre égal de noyaux à texture homogène choisis aléatoirement.
Le modèle de texture ainsi construit obtient un taux de bonne classification de 85.7% dans la phase d'apprentissage et 85.2% lors de la validation. Ce taux est bien inférieur à celui obtenu pour la forme[†]. Toutefois, il y a plusieurs explications à cet écart : la principale raison est le faible nombre d'individus appartenant à la classe "non homogène". Ce manque d'individu réduit les possibilités d'apprentissage. La deuxième raison est le biais introduit par le marqueur employé qui diminue la fiabilité du diagnostic.

5. Résultats, modèle complet : diagnostic des noyaux

Nous venons de construire deux modèles de classification qui caractérisent les deux principaux paramètres de diagnostic : la forme et la texture du noyau. Il faut maintenant combiner ces modèles afin de construire le modèle final qui prédit l'aspect anormal des noyaux. Ce dernier est une nouvelle combinaison linéaire des sept indices de formes du modèle de forme ainsi que des neuf coefficients du modèle de texture. Le taux de bonne classification obtenu par le modèle final est de 89.4% sur l'échantillon de validation.

6. Conclusions et perspectives

Nous venons de présenter une méthode de classement de noyaux de cellules de patients atteints par la maladie de la Progéria. Cette méthode repose essentiellement sur l'étude de la forme des noyaux à l'aide d'indices de formes. Dans un premier temps, nous avons construit des indices spécifiques au problème posé. Par la suite, ces indices ont parfaitement répondu au sous problème et ont apporté un taux de bon classement supérieur à 95%. Dans un souci de fiabilité et de validation, il n'est pas envisagé de chercher à améliorer ce résultat pour deux raisons : la première raison est qu'il faudrait apporter de nouvelles caractéristiques et que l'on risquerait de faire de l'apprentissage par cœur sur les données. Les quelques dixièmes de bonne classification gagnés lors des tests seraient perdus lors de la validation. La deuxième raison est que ce résultat est équivalent à celui produit en général par un expert.
Par la suite nous avons construit un modèle basé sur les caractéristiques Haralick afin de répondre au sous problème de caractérisation de la texture. Ce modèle n'a pas répondu de manière très satisfaisante au problème posé[‡] et n'a pas apporté d'amélioration probante dans le modèle final. En effet, le modèle de forme obtient à lui seul un taux de 88.9% de bonne classification pour le problème final. L'ajout du modèle de texture n'a permis d'améliorer le diagnostic que de 0.5%. Par conséquent, il nous faudra améliorer notre modèle afin de répondre de manière plus pertinente au sous problème de texture.
Le modèle final permet un taux de bonne classification proche de 90% en n'utilisant que des méthodes de classification linéaires dont la puissance est limitée. On pourrait améliorer ce taux en employant des méthodes moins contraintes telles que les réseaux de neurones, mais il est aussi utile de chercher à améliorer les caractéristiques extraites des noyaux pour récupérer plus d'informations pertinentes pour la classification.

Au vu des résultats apportés, il est évident qu'il nous faut améliorer le modèle de texture. Pour cela, nous souhaitons étendre la notion d'indices de formes et l'utiliser dans l'analyse de textures. La solution que nous souhaitons aborder pour y parvenir est de considérer une texture comme une carte de hauteur ; chaque pixel ne code plus un niveau de gris, mais une hauteur (image 8).

Nous souhaiterions alors extraire les caractéristiques de ce volume sous la nappe. Ces caractéristiques seraient essentiellement les variations d'altitude tels que les pics (figure 9), les crêtes, les lacs (figure 10) et les failles. Une fois extraites et isolées, ces informations sont de type volumique. Il serait alors possible d'utiliser des indices de formes 3D afin de les caractériser. Ces indices seraient soit les indices existant en 2D que nous étendrions en 3D, soit de nouveaux indices spécifiques au problème posé en 3D. Ces caractéristiques de volumes renseigneraient sur les variations d'intensité de la texture et permettraient de la classer.

[†] supérieur à 95%

[‡] 85.7% de bonne classification

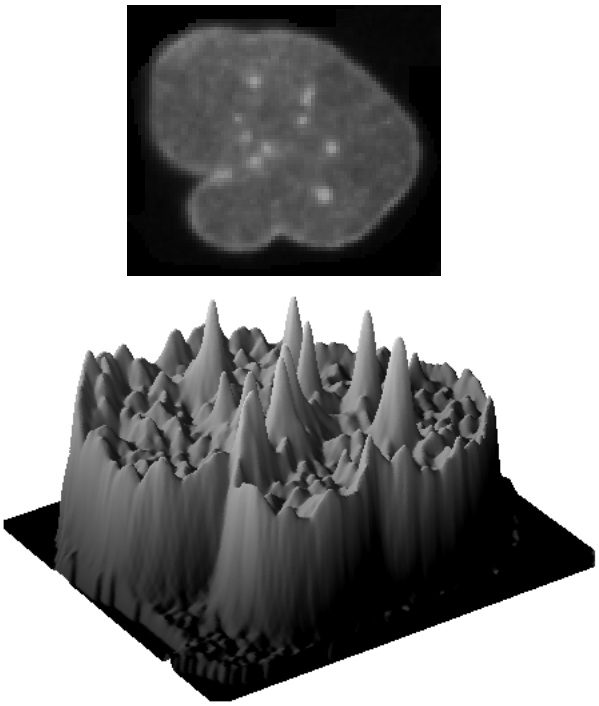


Figure 8: *Exemple d'un noyau de cellule et de son volume sous la nappe.*

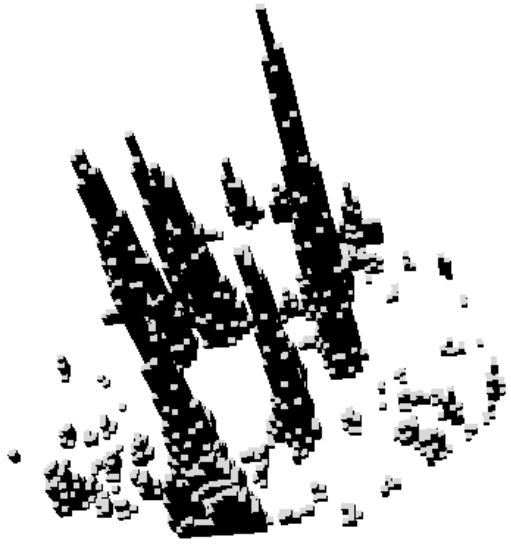


Figure 9: *Les pics extraient du volume sous la nappe de la figure 8.*

Figure 10: *Les lacs extraient du volume sous la nappe de la figure 8.*

References

[AAB07] ARAGON C. R., ARAGON D. B., BERKELEY L.: A fast contour descriptor algorithm for supernova image classification. In *Real-Time Image Processing 2007* (1 2007), vol. 6496.

[AGW97] AMIT Y., GEMAN D., WILDER K.: Joint induction of shape features and tree classifiers. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1997), vol. 19, IEEE.

[Ben02] BENDJOUDI H.: The gravelius compactness coefficient: critical analysis of a shape index for drainage basins. *Hydrological Sciences 47*, 6 (12 2002), 921–930.

[Bon20] BONNESEN T.: *Les problèmes des Iso-périmètres et des iséphanes*. Gauthier Villars, 1920.

[CC85] COSTER M., CHERMANT J.-L.: *Précis d'analyse d'images*. Editions du CNRS, 1985.

[CDT95] CHEN Y. Q., DIXON M. S., THOMAS D. W.: Statistical geometric features for texture classification. In *Pattern recognition* (1995), vol. 28, pp. 537–552.

[Cos01] COSTA J.-P. D.: *Analyse statistiques de textures directionnelles*. PhD thesis, Université Bordeaux I, 12 2001.

[DS89] D.W. H., S. L.: *Applied Logistic Regression*. John Wiley & Sons, Toronto, 1989.

[Fil95] FILLERE I.: *Outils mathématiques pour la reconnaissance de formes*. PhD thesis, Université de St Etienne, Septembre 1995.

[Gär99] GÄRTNER B.: Fast and robust smallest enclosing ball. In *ESA '99: Proceedings of the 7th Annual European Symposium on Algorithms* (London, U, 1999), vol. 1643, Springer-Verla, pp. 325–338.

[Har79] HARALICK R. M.: Statistical and structural approaches to texture. In *Proceedings of the IEEE* (1979), vol. 67, pp. 786–804.

[HSD73] HARALICK R. M., SHANMUGAM K., DINSTEIN I.: Textural features for image classification. In *IEEE Transactions on Systems, Man and Cybernetics* (1973), vol. 3, pp. 610–621.

[IP97] IIVARINEN J., PEURA M.: Efficiency of simple shape descriptors. In *International Workshop on Visual Form* (5 1997).

[IPSV97] IIVARINEN J., PEURA M., SÄRELÄ J., VISA A.: Comparison of combined shape descriptors for irregular objects. In *8th British Machine Vision Conference, BMVC'97* (Essex, Great Britain, 1997), vol. 2, pp. 430–439.

[KR06] KPALMA K., RONSIN J.: Multiscale contour description for pattern recognition. In *IEEE Transactions on Pattern Recognition Letters* (2006), IEEE, pp. 1545–1559.

[Mav01] MAVROMATIS S.: *Analyse de texture et Visualisation scientifique*. PhD thesis, Université de la Méditerranée, 2001.

[San76] SANTALO L.: *Integral Geometry and Geometric Probability*. Addison Wesley, 1976.

[SEB*03] SBONER A., ECCHER C., BLANZIERI E., BAUER P., CRISTOFOLINI M., ZUMIANI G., FORTI S.: A multiple classifier system for early melanoma diagnosis. *Published in Artificial Intelligence in Medicine 27*, 1 (2003), 29–44.

[SGBC*03] SANDRE-GIOVANNOLI A. D., BERNARD R., CAU P., NAVARRO C., AMIEL J., BOCCACCIO I., LYONNET S., STEWART C. L., MUNNICH A., MERRER M. L., LEVY N.: Lamin a truncation in progeria, 2003.

[SR04] SOLTANZADEH H., RAHMATI M.: Recognition of persian handwritten digits using image profiles of multiple orientations. In *IEEE Transactions on Pattern Recognition Letters* (2004), IEEE, pp. 1569–1576.

[THK06] TOUMI A., HOELTZENER B., KHENCHAF A.: Classification des images isar pour la reconnaissance des cibles. In *XIIIème Rencontres de la Société Francophone de Classification (SFC)* (2006).

[TLG*03] TUSET V. M., LOZANO I. J., GONZÁLEZ J. A., PERTUSA J. F., GARCÍA-DÍA M. M.: Shape indices to identify regional differences in otolith morphology of comber. In *Journal of Applied Ichthyology* (4 2003), vol. 19, p. 88.

[URW*02] UNG A., RANCHIN T., WALD L., WEBER C., HIRSCH J., PERRON G., KLEINPETER J.: Cartographie de la pollution de l'air: une nouvelle approche basée sur la télédétection et les bases de données géographiques. application à la ville de Strasbourg. In *Journées CASSINI 2002 du GDR CASSINI-SIGMA* (September 2002).

[ZR04] ZUNIC J., ROSIN P. L.: A new convexity measure for polygons. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2004), vol. 26, pp. 923–934.

Annexe : liste des indices de formes utilisés dans cet article

$$Allongement_{Axes\ principaux} = \frac{L_{AP2}}{L_{AP1}} \in [0,1]$$

$$Allongement_{Diametre} = \frac{E}{D} \in [0,1]$$

$$Allongement_{Rayons} = \frac{\rho_i}{\rho_e} \in [0,1]$$

$$Allongement_{Geodesique} = \frac{4}{\pi}\frac{A}{D_G^2} \in]0,1]$$

$$Circularite = \frac{R_{min}}{R_{max}} \in [0,1]$$

$$Circularite_{Courbure} = \frac{\rho_{min}}{\rho_{max}}$$

$$Convexite_{Perimetrique} = \frac{P(ConvexHull(F))}{P(F)}$$

$$Convexite_{Surfacique} = \frac{A(F)}{A(ConvexHull(F))}$$

$$Deficit = 1 - \pi\frac{(\rho_e - \rho_i)^2}{P^2} \in [1 - \frac{\pi^2}{16}, 1]$$

$$Deficit_{IsoPerimetrique} = 4\pi\frac{A}{P^2} \in]0,1]$$

$$Ecart\ disque\ inscrit = \frac{\pi\rho_i^2}{A} \in [0,1]$$

$$Etalement_{Morton} = \frac{4}{\pi}\frac{A}{L_{AP}^2} \in]0,1]$$

$$Symetrie_{Besicovitch} = \sup_{x\in F}\frac{A(F \cap Symetrique(F,x))}{A(F)}$$

$$Variance\ circulaire = \frac{1}{S\mu_r^2}\sum_{p_i\in F}(\|p_i - B\| - \mu_r)^2$$

$$\Psi_{Ellipse} = \frac{\pi R_{min}R_{max}}{A}\ ou\ \frac{\pi}{4}\frac{L_{AP1}L_{AP2}}{A} \in [0,1]$$

$$\Psi_{N_{Cce}} = \frac{1}{1+N_{Cce}} \in]0,1]$$

$$\Psi_{Parallelogramme} = \frac{A}{E_{AP}L_{AP}}\ ou\ \frac{A}{E\times D}$$

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Visualisation Focus+Contexte pour l'Exploration Interactive de Maillages Tétraédriques

Sébastien Barbier [1] et Georges-Pierre Bonneau[1]

[1]Grenoble Universités - INRIA - LJK

Abstract

L'exploration et l'analyse visuelle de grands maillages tétraédriques restent des tâches coûteuses en temps lorsque les ensembles de données sont affichés dans leur globalité. Pourtant, dans la plupart des cas, l'utilisateur n'explorera que de petites zones compactes où se concentrent les informations qu'il juge remarquables. Se basant sur ce constat, nous proposons une approche focus+contexte reposant sur une double résolution des données. Dans l'espace objet, une Région Locale d'Intérêt (RLI) - le focus - est extraite du maillage précis originel et est entourée par une représentation grossière globale - le contexte. Pour unir les deux résolutions, une connexion topologiquement valide est créée interactivement. Les techniques de rendu classiques y sont intégrées.
De plus, quand le focus est déplacé, l'extraction de la RLI ainsi que son affichage sont accélérés en utilisant la cohérence temporelle. Les dernières cartes graphiques sont utilisées afin d'accélérer le Rendu Volumique Direct. Notre approche focus+contexte réduit de manière significative le nombre de primitives affichées ce qui permet une exploration interactive de grands maillages tétraédriques.

1. Introduction

Les simulations numériques de phénomènes naturels ou pour l'ingénierie sont réalisées sur des maillages composés au moins de millions de cellules. Le champ scalaire résultant est analysé visuellement mais l'affichage global de tels ensembles de données reste un processus coûteux en temps. Cependant, un rendu global précis est rarement nécessaire car l'utilisateur concentre souvent son attention sur des zones remarquables dans lesquelles une grande précision est essentielle alors qu'une simplification peut être réalisée à l'extérieur pour conserver un contexte informatif. De plus, il est capital de pouvoir explorer interactivement les alentours de la région d'intérêt afin de la situer pleinement dans son contexte.

La majorité des approches précédentes se basent sur une structure multirésolution pour n'afficher que les sous-parties visibles à la meilleure précision. Au cours d'une étape de précalcul, des structures de données multirésolution sont extraites via un algorithme de simplification. Au cours de l'exécution, la résolution est adaptée ce qui permet de réduire le nombre de tétraèdres, donc d'accélérer la phase de rendu.

Plus récemment, des algorithmes de focus+contexte monorésolution ont été proposés, restreignant le focus selon les besoins de l'utilisateur. La plupart d'entre eux ont été implémentés sur des grilles régulières ou s'appuient sur une extraction haut-niveau de l'information.

Dans cet article, nous proposons un algorithme pour des maillages tétraédriques qui mélange l'approche focus+contexte avec une structure bi-résolution. Le focus ou Région Locale d'Intérêt (RLI) peut être déplacé interactivement n'importe où dans l'ensemble de données, en particulier dans les zones remarquables. Une résolution grossière entoure la RLI comme contexte afin de guider l'exploration. Les techniques de rendu classiques : extraction d'isosurfaces, plans de coupe texturés ou encore Rendu Volumique Direct (DVR); sont pleinement intégrés.

Les principales contributions sont les suivantes :

- Une approche focus+contexte efficace et centrée sur l'utilisateur reposant sur une structure à deux résolutions

pour explorer interactivement de grands maillages tétraédriques. Une telle approche permet de réduire les temps de précalculs tout en simplifiant les structures de données par rapport aux structures multirésolution.
- L'utilisation de la cohérence temporelle pour accélerer le déplacement et le rendu du focus au sein des zones remarquables.
- Des améliorations de méthodes existantes pour le DVR reposant sur l'exploitation des dernières fonctionnalités des cartes graphiques. Le tri des tétraèdres est en partie réalisé sur GPU via un algorithme de depth-peeling et un geometry shader permet d'accélerer la méthode de projection des primitives.

La suite de cet article est organisée de la manière suivante : la section 2 est un bref état de l'art sur les précédentes approches. Notre structure bi-résolution est détaillée dans la section 3 et son accélération via la cohérence temporelle dans la section 4. Les améliorations sur le pipeline graphique concernant le DVR sont décrites dans la section 5. Des résultats sont fournis dans la section 6 et nous concluons dans la dernière section.

2. État de l'Art

2.1. Simplification et Multirésolution

Afin de visualiser de grands ensembles de données, des algorithmes de simplification et multirésolution ont été proposés pour diminuer le nombre de tétraèdres affichés. Une majorité d'entre eux reposent sur des opérations de contractions d'arêtes. La simplification se base sur un ordonnancement des arêtes au sein d'une pile par rapport à l'évaluation d'une somme pondérée d'une métrique d'erreur scalaire [THJ99, GVW99] qui permet de préserver les variations du champ scalaire; et d'une métrique d'erreur de domaine [CCM*00, NE04] qui interdit des contractions qui causeraient une grande déformation de l'aspect global du maillage. Avant chaque contraction, celle-ci est testée afin d'éviter toute modification topologique ou géométrique qui rendrait le maillage simplifié incorrect.

Les structures multiresolution sont construites en aval de ces simplifications afin d'afficher dynamiquement des maillages tétraèdriques dépendant du point de vue. Staadt et Gross [SG98], Cignoni et al. [CFM*04] et Sondershaus et Strasser [SS05] utilisent des forêts d'arbres binaires pour stocker les dépendances entre les contractions de manière compacte. Callahan et al [CCSS05] généralisent leur système nommé HAVS pour afficher des niveaux de détails globaux. Sondershaus et Strasser [SS06] développent une segmentation des ensembles de données. Un octree guide la construction d'une hiérarchie multirésolution stockée dans un graphe direct sans cycle (DAG).

Cignoni et al. [CMS94] introduit un outil local, la *MagicSphere*, à l'intérieur de laquelle ils appliquent un filtre multirésolution pour l'extraction d'isosurfaces. A chaque étape, une région sphérique est définie dans l'espace objet et une isosurface est recalculée en parcourant tout le maillage. Aucune connexion entre les deux résolutions n'est construite, mais une transition douce est simulée via l'utilisation du alpha buffer.

2.2. Approches Focus+Contexte

Récemment, des approches focus+contexte monorésolution ont été proposées pour visualiser les volumes de données selon les besoins de l'utilisateur.

Les méthodes focus+contexte les plus populaires sont élaborées pour des données médicales régulières segmentées. Hadwiger et al. [HBH03] proposent d'appliquer différentes techniques de rendu pour mettre en évidence les informations remarquables. Wang et al. [WZMK05] définissent une *Magic Volume Lens* qui agrandit dans l'espace image les données qu'elle contient. Viola et al. [VFSG06] proposent des solutions pour déterminer automatiquement les zones remarquables.

Pour les maillages tétraédriques, les méthodes focus+contexte existantes [DGH03, PKH04] mélangent des techniques de rendu global avec une représentation 2D des données scalaires afin de localiser les zones remarquables dans l'espace et dans le temps.

2.3. Rendu Volumique Direct

Le Rendu Volumique Direct est une technique de rendu pour visualiser des maillages volumiques. Nous rappelons certains algorithmes permettant de la réaliser.

Les méthodes de Ray-Casting ont été implémentées sur GPU [WKME03]. Mais elles restent contraintes par la taille mémoire des cartes graphiques ce qui ne permet pas de visualiser de grands ensembles de données.

Les techniques de projection de cellules réalisent l'intégration des rayons en déterminant la projection des tétraèdres dans l'espace image [ST90, WMFC02, MMFE06]. Ces méthodes nécessitent en amont un tri des cellules selon leur visibilité [CMSW04].

Le système *HAVS* [CICS05] est une approche hybride qui exécute deux tris consécutifs sur les faces des tétraèdres : le premier calcule un ordre partiel sur le CPU et le second un ordre total sur le GPU. Mais les limitations actuelles du GPU peuvent produire des tris locaux incorrects.

3. Focus+Contexte via une structure birésolution

Nous proposons une approche focus+context pour des maillages tétraédriques reposant sur des résolutions fine et grossière des données. Une telle stucture birésolution a tout d'abord été introduite par Cignoni et al. [CMS94] mais uniquement pour l'extraction d'isosurfaces. Nous proposons

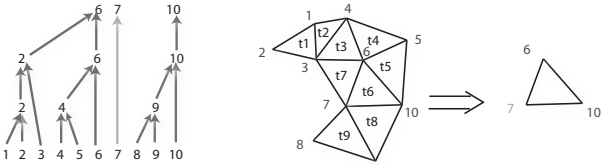


Figure 1: *Processus de Simplification. Les sommets sont regroupés en 3 clusters C_0^v={1,2,3,4,5,6}, C_1^v={7} et C_2^v={8,9,10}; et les triangles dans C_0^t={$t_1,t_2,t_3,t_4,t_5,t_6,t_7$}, C_1^t={t_6,t_7,t_8,t_9} et C_2^t={t_5,t_6,t_8,t_9}.*

Figure 2: *Création du maillage grossier à partir de la clusterisation. Le triangle sombre appartient à trois clusters (délimités par les traits épais). Les sommets fins sont projetés sur les grossiers transformant le triangle fin sombre en un triangle grossier clair.*

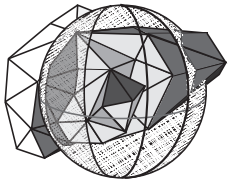


Figure 3: *La RLI est composée de tous les tétraèdres contenus dans la sphère. Tous les tétraèdres colorés sont internes. Les gris clairs sont complets, les foncés sont de bord. Le tétraèdre central est le tétraèdre racine.*

une nouvelle approche permettant de construire une connexion valide entre les deux résolutions de manière efficace en temps pour obtenir un unique maillage et utilisant la cohérence temporelle. Contrairement aux approches multirésolution, aucune structure de données complexe n'est nécessaire. De plus, toutes les techniques de rendu classiques sont pleinement intégrées.

Nous détaillons ci-après les étapes de la construction de notre approche. L'algorithme est illustré par des schémas sur des surfaces dans un souci de clarté.

3.1. Contruction du maillage grossier

Un algorithme de simplification similaire à [CCM*00] est exécuté en préprocessing. Il construit une forêt d'arbres binaires où les feuilles sont les sommets fins initiaux et les racines les sommets grossiers finaux. L'utilisation de *half-edge collapses* n'introduisant pas de nouveaux sommets, les sommets grossiers appartiennent au maillage initial. Ainsi, une clusterisation du maillage initial est calculée : les sommets fins qui s'effondrent sur le même sommet grossier sont regroupés en un cluster (cf. Figure 1).

Chaque sommet fin conserve l'identité de son cluster. Chaque tétraèdre fin est associé avec un à quatre clusters en fonction du regroupement de ses sommets. Le maillage grossier est déduit de la clusterisation et du maillage fin. Seuls les tétraèdres appartenant à quatre clusters distincts sont conservés et transformés en projetant leurs sommets fins sur les sommets grossiers correspondants (cf Figure 2).

3.2. Extraction de la Région Locale d'Intérêt

La RLI est construite dans l'espace objet à partir d'une sphère dont le centre et le diamètre sont définis par l'utilisateur. La RLI consiste en l'ensemble des tétraèdres fins contenues à l'intérieur de cette sphère. Pour localiser rapidement le centre d'intérêt au sein du maillage fin, un octree est construit sur les barycentres de tous les tétraèdres. Le tétraèdre le plus proche du centre est alors considéré comme le centre de la RLI et est appelé *racine*. Dans le but de profiter de la cohérence temporelle dans le cas d'une exploration, la recherche recommence toujours à partir du dernier noeud trouvé. Dans tous les cas, la localisation du tétraèdre *racine* est négligeable devant le reste du pipeline graphique.

En partant du tétraèdre *racine* et en utilisant les relations d'adjacence, la RLI est agrandie via un parcours en profondeur jusqu'à ce que tous les tétraèdres contenus dans la sphère virtuelle soient ajoutés. Parmi ces tétraèdres, on appelle *interne de bord* ceux intersectant la sphère, *interne complet* les autres (cf Figure 3). Les tétraèdres interne complet sont stockés dans une table de hachage et les internes de bord dans une liste dite *de bord* afin d'utiliser judicieusement la cohérence temporelle - cf section 4.1.

Afin de ne pas omettre des composantes connexes, des parcours supplémentaires sont réalisés.

Pour plonger la RLI au sein du contexte, les clusters intersectant la RLI et leurs sommets grossiers associés sont marqués *actifs*. Les sommets et les tétraèdres fins appartenant à un cluster *actif* sont aussi appelés *actifs*. De plus, les tétraèdres fins ayant quatre sommets actifs sont nommés *intérieurs* sinon *de lien*.

Le contexte est défini par tous les tétraèdres grossiers qui n'intersectent pas le focus. Deux composantes connexes sont ainsi obtenues.

3.3. Lien entre les deux résolutions

Pour obtenir un unique maillage valide, une connexion doit être construite entre le focus et le contexte. Cette transition

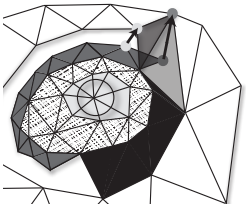


Figure 4: *Le raccord est en cours de construction. Les triangles fins de lien (gris foncé) sont étirés (en noir) pour relier les deux résolutions et combler le trou. Cela est fait en projetant les sommets fins inactifs sur leur sommet grossier correspondant. Le focus est représenté par le cercle. Les triangles gris clairs et hachurés sont internes et intérieurs, respectivement, alors que les blancs font partie du contexte.*

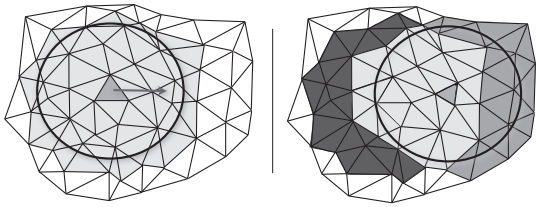


Figure 5: *Application de la cohérence temporelle. La primitive sombre centrale est déplacée vers la droite. Les primitives enlevées durant la mise à jour sont en noir, alors que les nouvelles sont en gris sombre.*

doit être continue entre les deux résolutions. Ainsi, le focus sera entouré par le contexte sans perte d'informations qui pourraient se révéler essentielle à la compréhension globale de la simulation.

Pour raccorder la RLI au maillage grossier, des tétraèdres sont ajoutés. Afin d'assurer une robustesse topologique, les heuristiques reposant sur la géométrie sont évitées. Nous proposons une solution s'appuyant sur la clusterisation, à la fois rapide et valide dans toutes les situations.

Un unique maillage valide est ainsi obtenu, composé de tétraèdres grossiers, fins et étirés :

- Tétraèdres grossiers : tous les tétraèdres ayant quatre sommets inactifs;
- Tétraèdres fins : tous les tétraèdres actifs intérieurs, incluant ceux contenus dans la RLI (tétraèdres internes);
- Tétraèdres étirés : tous les tétraèdres actifs de lien sont étirés en projetant les sommets fins inactifs sur leurs sommets grossiers associés. Cela peut produire des tétraèdres dégénérés qui sont enlevés au cas par cas - sauf pour le DVR.

La Figure 4 illustre la construction de ce maillage unique. Notre solution augmente le nombre de tétraèdres affichés mais en contrepartie assure une construction rapide et simple. De plus, le lien est topologiquement correct.

4. Cohérence Temporelle

Les techniques de rendu telles que les plans de coupe texturés, l'extraction d'isosurfaces ou le DVR sont intégrés au sein de notre approche focus+contexte. Pour accélérer l'ensemble du pipeline graphique, l'extraction de la RLI et les techniques de rendu utilisent la cohérence temporelle lors d'un déplacement du focus.

4.1. Mise à jour de la Région Locale d'Intérêt

Lorsque l'utilisateur déplace le centre d'intérêt lors d'une exploration, un nouveau tétraèdre *racine* est recherché. Si celui-ci n'appartient pas à l'ancienne RLI, la table de hachage et la liste de bord sont effacées et une nouvelle région est reconstruite comme expliqué en section 3.2.

Sinon, la nouvelle RLI est mise à jour par rapport à l'ancienne en trois étapes, comme illustré en Figure 5 :

1. Nettoyer la liste de bord en enlevant les tétraèdres qui ne le sont plus. Ceux-ci sont stockés dans une liste extérieure;
2. Parcourir la liste extérieure et utiliser les relations d'adjacence pour enlever tous les tétraèdres n'appartenant plus à la nouvelle RLI et mettre à jour la liste de bord.
3. Ajouter les nouveaux tétraèdres en partant de la nouvelle liste de bord en parcourant à nouveau les tétraèdres via les relations d'adjacence. Les internes sont stockés dans la table de hachage, les tétraèdres de bord dans la liste de bord.

La table de hachage assure une insertion et une suppression en temps constant. La complexité globale de la mise à jour est donc linéaire en nombre de tétraèdres enlevés puis ajoutés. Cette complexité est négligeable quand l'utilisateur déplace le focus avec de petits déplacements, ce qui est le cas lors de l'exploration des alentours des zones remarquables.

4.2. Isosurfaces

Un *Marching Tetrahedra* permet d'extraire les isosurfaces. L'isosurface grossière est calculée une fois par isovaleur. Par contre, l'isosurface fine est extraite des tétraèdres intérieurs et de lien et doit être mise à jour lorsque le focus est déplacé.

Pour cela, la cohérence temporelle minimise le nombre d'extractions. Quand un cluster devient actif, l'isosurface correspondante est calculée une fois pour tous ses tétraèdres et sauvegardée. Ainsi, lorsque la RLI est déplacée mais que le cluster reste actif, aucune nouvelle extraction n'est faite.

4.3. Rendu Volumique Direct

Le DVR est intégré au sein de notre structure bi-résolution. Afin d'éviter les limitations dues à la mémoire ou à des opérations de lecture/écriture incertaines des cartes graphiques, un ordre de visibilité sur CPU : *SXMPVO* [CMSW04] et une technique de splatting sur GPU : GATOR [WMFC02] sont implémentés.

Nous rappelons dans cette section les étapes de l'algorithme SXMPVO. Puis, nous détaillons les transformations faites pour intégrer la cohérence temporelle et le DVR au sein de notre approche.

4.3.1. SXMPVO

L'algorithme SXMPVO garantit un ordre total de visibilité pour les maillages non convexes et non connexes sans cycles. Il est composé de quatre étapes. L'étape I détermine un ordre partiel de visibilité en utilisant les relations d'adjacence dans les zones connexes. L'étape II trie les faces de bord selon leur profondeur et les envoie à l'étape III qui détermine leur ordre de visibilité dans l'espace objet via un A-Buffer. Enfin, l'étape IV réalise un parcours en profondeur à partir des faces non cachées afin d'afficher en utilisant le GPU l'ensemble des primitives de l'arrière vers l'avant.

4.3.2. SXMPVO Cohérent

Le DVR étant dépendant du point de vue, l'ordre de visibilité doit être recalculé pour toutes les primitives de la RLI si celui-ci est modifié. Aucune amélioration ne peut être alors envisagée. Dans le reste de cette section, on suppose que l'utilisateur a fixé son point de vue et déplace uniquement le focus.

L'étape I de SXMPVO détermine un ordre partiel en se basant sur les relations d'adjacence. Lors d'une exploration où le point de vue n'est pas modifié, cet ordre de visibilité est inchangé. Ainsi, seuls les tétraèdres nouvellement ajoutés doivent être insérés dans le tri. Cette insertion est faite lors de l'étape 3 de la mise à jour de la RLI - cf section 3.2. De plus, lorsque les tétraèdres sont enlevés de la RLI, leurs relations de visibilité sont effacées pour ne pas être envoyés au GPU lors du parcours en profondeur.

Cette mise en place de la cohérence temporelle permet de réduire la complexité de l'étape I au nombre de tétraèdres ajoutés et enlevés lors de la mise à jour.

L'étape IV est modifiée pour afficher l'ensemble des tétraèdres de la RLI. Durant le parcours en profondeur, les primitives sont marquées pour éviter des rendus multiples ou détecter des cycles de visibilité. Le test de visite a été modifié afin de connaître durant quel rendu le tétraèdre a été marqué. Si la marque correspond au rendu actuel, le parcours est arrêté, sinon il correspond au rendu précédent et le tétraèdre est parcouru. Cette modification assure que tous les tétraèdres et en particulier les anciens, sont envoyés au GPU.

4.3.3. Intégration du Rendu Volumique Direct

Dans notre structure focus+contexte, le DVR peut être appliqué à la RLI, sur les deux résolutions ou sur le maillage unique valide.

Si le point de vue est fixe, le SXMPVO cohérent est appliqué au sein de la RLI. Dans le cas des deux résolutions, SXMPVO est appliqué sur les deux composantes connexes.

Afficher le maillage unique valide avec le DVR implique un stockage en mémoire de l'ensemble des cellules de lien alors qu'elles sont déduites par projection des sommets inactifs pour les autres techniques. En effet, pour le DVR les relations d'adjacence entre les cellules de lien sont nécessaires pour assurer un ordre partiel correct en étape I, ce qui implique leur mémorisation. Elles sont mises à jour à chaque fois que la RLI est déplacée.

Cette mise à jour se déroule pendant l'étape I :

1. **Trouver les tétraèdres de lien**. Les tétraèdres de lien sont sauvegardés dans une table de hachage. Pour les trouver, l'ensemble des tétraèdres grossiers et des tétraèdres fins actifs sont testés sur leur statut. Tous les tétraèdres de lien détectés sont stockés, dégénérés ou non. Les relations d'adjacence avec les tétraèdres fins et grossiers sont calculées.
2. **Mettre à jour les relations d'adjacences entre cellules de lien**. Pour cela, les relations d'adjacence entre les tétraèdres fins correspondant sont utilisés. En effet, chaque cellule de lien est en fait une cellule fine étirée. Ces cellules fines sont donc visitées jusqu'à trouver une cellule fine dont la cellule de lien correspondante n'est pas dégénérée. A la fin de cette étape, les cellules de lien dégénérées sont enlevées.

A la fin de l'étape I, un ordre partiel est ainsi obtenu pour le maillage unique valide.

5. Améliorations dans le Rendu Volumique Direct

Les implémentations originales de SXMPVO et de GATOR ont certaines limitations : le A-Buffer (étape III) de SXMPVO est calculé sur le CPU et GATOR réalise des calculs redondants sur le GPU. Nous proposons des améliorations afin de pallier à ces inconvénients.

5.1. Depth-Peeling

Pour accélérer SXMPVO, une implémentation sur GPU de l'algorithme de *depth-peeling* est implémenté pour remplacer le A-Buffer. Cette idée a été citée mais jamais implémentée pour SXMPVO.

Le depth-peeling réalise des rendus successifs de la scène tout en enlevant à chaque étape les faces visibles du rendu précédent. En envoyant au GPU uniquement les faces de bord déduites lors de l'étape I, on peut ainsi obtenir au sein de textures l'ensemble des relations de visibilité dans l'espace image.

5.2. Geometry Shader

L'algorithme GATOR est la première méthode de projection de tétraèdres implémentée sur GPU. Un vertex shader projette les cellules sur un graphe de base dans l'espace image. Chaque projection est faite cinq fois par tétraèdre pour garantir une transmission sans perte au fragment shader. Cette opération est donc redondante et trop de données sont envoyées au GPU.

Pour améliorer cette approche, le vertex shader est transformé en geometry shader. Cela permet certaines améliorations. Un tétraèdre est envoyé une seule fois au lieu de cinq fois sous la forme d'une ligne avec adjacence de quatre points repérés par leur position et leur couleur. La taille des données envoyée au GPU est donc grandement diminuée. Chaque tétraèdre est projeté une seule fois au sein du geometry shader via quelques modifications de l'implémentation originale. De plus, aucune primitive dégénérée n'est envoyée au fragment shader. Cette solution évite les deux passes que proposaient Marroquim [MMFE06].

6. Résultats

Nous avons implémenté nos résultats sur un ordinateur Windows XP 32 bits 2,8 GHz, 2 Go de RAM avec une GeForce 8800 GTS de 640 Mo de mémoire vidéo. Nous avons testé notre algorithme sur des ensembles de données listés dans le Tableau 1. L'ensemble DTI est un maillage médical régulier que nous avons convertit en tétraèdres.

6.1. Résultats Visuels

Des résultats visuels de notre approche sont visibles en Figures 6, 8 et 9. Pour mettre en évidence la RLI, une fonction de transfert en couleurs est appliquée au sein du focus et en niveau de gris au sein du contexte. De plus, l'aspect découpé de la RLI pour le DVR est adouci via l'utilisation d'un stencil buffer.

Notre méthode peut, dans certains cas, produire des flips et des tétraèdres slivers au sein du lien, cause de certains artefacts visuels. Cependant, ils sont vraiment limités puisque sur l'ensemble de nos tests moins de 0,01% des tétraèdres étirés s'inversent et moins de 7% ont un ratio d'aspect supérieur à 20.

La Figure 6 montre l'extraction d'une isosurface au sein du Post. La RLI est déplacée au plus près du tube pour comprendre l'écoulement de l'oxygène liquide. La construction du lien entre les deux résolutions apportent clairement les informations manquantes dues au trou et cela permet une meilleure compréhension du phénomène par rapport à un simple mélange des résolutions au niveau du raccord via un alpha-blending.

La Figure 8 illustre le DVR appliqué au Blunt. La RLI est placée dans la zone de turbulence, où la densité de l'air varie le plus fortement. Déplacer la RLI le long de l'arête permet de comprendre son influence sur les variations du flux d'air. L'analyse peut être affinée grâce à un rendu global.

Nom	sommets	tétraèdres	champ scalaire
DTI	950,272	4,596,765	$[1, 203]$
Bucky	262,144	1,250,235	$[0, 254]$
Post	108,300	616,050	$[-0.54, 4.4]$
Blunt	31,858	222,414	$[0.19, 4.98]$

Table 1: *Ensembles de données utilisés par les résultats.*

Les différentes techniques de rendu sont rassemblées en Figure 9. L'utilisateur recherche l'hypotalamus rendu en DVR. Il déplace sa RLI en se localisant grâce au plan de coupe texturé et à l'extraction de l'isosurface. La RLI raffine aussi l'isosurface.

6.2. Temps de Rendu

Pour valider nos améliorations de l'étape III de SXMPVO et du pipeline de GATOR, une RLI composée de 121 159 tétraèdres est affichée au sein du Blunt afin de founir une complexité pertinente. Les résultats pour les isosurfaces et le DVR sont calculés avec le même déplacement du centre d'intérêt au sein du Bucky correspondant à l'exploration des atomes de carbone. Nous avons testé notre approche sur les autres ensembles et les temps de rendu sont similaires.

6.2.1. Depth-Peeling et Geometry Shader

La Table 2 illustre l'amélioration en temps obtenue grâce à l'utilisation du depth-peeling (III) et du geometry shader (IV). Aucune cohérence temporelle n'est utilisée, différentes résolutions d'images sont essayées.

Alors que le nombre de pixels augmente, notre nouvelle phase III devient de plus en plus rapide par rapport à l'ancienne. Zoomer sur la RLI ralentit peu cette phase.

Le geometry shader est implémenté avec la table de projection de GATOR. Nous avons aussi essayé celle de Marroquim mais les temps de rendu sont plus lents (63 au lieu de 55 ms) sans différences visuelles remarquables. Les temps globaux de rendu sont divisés en moyenne par 2,5 par rapport à l'approche originale.

6.2.2. Isosurfaces

La Table 3 compare l'extraction d'isosurfaces avec ou sans la cohérence temporelle moyennant plusieurs parcours avec différentes tailles de RLI. Avec la cohérence temporelle, ce débit est multiplié par 2,6.

6.2.3. Rendu Volumique Direct

La Figure 7 montre le nombre d'images par seconde pour le DVR en utilisant le depth-peeling et le geometry shader dans une fenêtre 680x840.

Taille Image	SXMPVO Original				SXMPVO Amélioré			
680x840	I+ II	III	IV	R	I	III	IV	R
0,3%	195	24	139	2,75	188	31	54	3,5
2,3%	257	87	138	2,1	185	33	55	3,5
31,1%	381	476	138	1	185	42	55	3,4
99,9%	242	1629	139	0,5	187	66	56	3,1
945x1210	I+ II	III	IV	R	I	III	IV	R
2,1%	296	122	139	1,9	187	64	55	3,2
29,3%	338	1173	137	0,7	189	78	56	3,0
99,9%	204	2962	134	0,3	187	126	55	2,6

Table 2: *Comparaisons du temps de rendu pour le DVR pour deux résolutions d'image. 121 259 tétraèdres sont affichés. Les temps sont exprimés en millisecondes pour les étapes (I,II,III,IV) de SXMPVO. Le rendu global (R) est en fps. Le pourcentage de l'image occupé par la projection de la RLI est indiqué.*

	Sans	Avec
Extraction	250.000 tet/s	650.000 tet/s
Rendu	500.000 tet/s	500.000 tet/s

Table 3: *Nombres de tétraèdres extraits avec ou sans la cohérence temporelle, moyenne sur plusieurs tailles de RLI.*

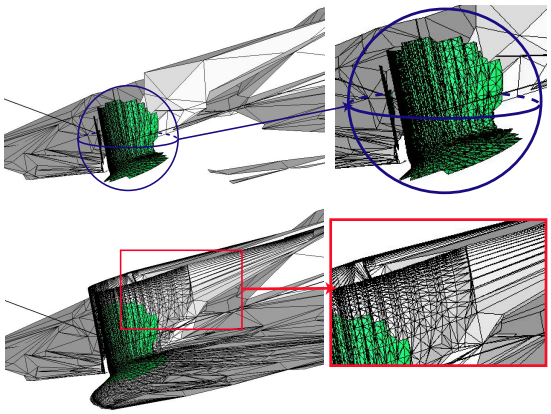


Figure 6: *Extraction d'isosurfaces au sein du Post pour la RLI seulement (en haut) et avec un unique maillage valide (en bas). La sphère d'intérêt est dessinée en haut à gauche, zoomée en haut à droite . La RLI met en évidence l'écoulement de l'oxygène liquide.*

Dans un souci de clarté, nous distinguons les algorithmes original et cohérent. Nous séparons aussi le rendu avec et sans le lien. Un rendu avec le maillage unique est plus approprié pour une analyse des zones locales remarquables alors que le rendu de la RLI en DVR avec d'autres rendus appliqués au contexte l'est plus pour une localisation ou une exploration globale.

Nos temps de rendu quand le maillage unique est utilisé sont similaires voire meilleurs aux dernières approches multirésolution, mais notre technique assure un temps de prépocessing moins important, des structures de données plus simples et une extraction de la zone de focus à n'importe quel endroit dans l'ensemble de données. On peut de plus remarquer que :

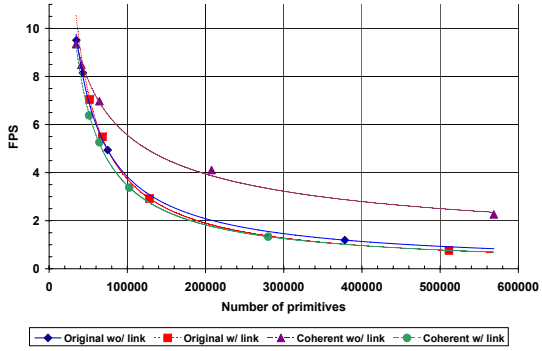


Figure 7: *Images par seconde obtenues via notre approche par rapport au nombre de tétraèdres affichés via un DVR.*

- La cohérence temporelle accélère le temps d'affichage global quand le focus est rendu sans le lien. Cette situation est bien adaptée à une exploration rapide des maillages. Ainsi, un déplacement interactif est garanti pour localiser les zones remarquables.
- La contruction et la mise à jour du lien introduisent de légers surcoûts en étape I mais qui sont sans réels impacts sur les temps globaux de rendu.

Notre approche focus+contexte assure des temps de rendu interactifs si la RLI conserve une taille raisonnable par rapport à la taille de l'ensemble de données. Ce n'est pas une contrainte forte car la majorité des informations remarquables se trouvent dans des zones compactes. Une RLI de diamètre égale à 10-20% de la diagonale de la boîte englobante du maillage semble un bon choix qui garantit 3 à 9 images par seconde avec ou sans la construction du lien. Un plus grand focus peut être même défini interactivement sans la construction du lien.

7. Conclusion

Nous venons de présenter une approche focus+contexte permettant d'explorer interactivement de grands ensembles de données tétraédriques reposant sur une structure bi-résolution. Une RLI est extraite du maillage fin dans l'espace objet et est entourée par une représentation grossière qui définit le contexte. La clusterisation construite lors d'un traitement en préprocessing permet d'extraire rapidement un maillage unique lors de l'exécution en unifiant la RLI avec son contexte. Les techniques de rendu ont été intégrées dans notre approche. La cohérence temporelle ainsi que les dernières fonctionnalités des cartes graphiques sont pleinement exploitées pour assurer une interactivité de l'exploration.

References

[CCM*00] CIGNONI P., CONSTANZA D., MONTANI C., ROCCHINI C., SCOPIGNO R.: Simplification of tetrahedral meshes with accurate error evaluation. In *VIS '00: Proceedings of the conference on Visualization '00* (2000), pp. 85–92.

[CCSS05] CALLAHAN S., COMBA J., SHIRLEY P., SILVA C.: Interactive rendering of large unstructured grids using dynamic level-of-detail. In *IEEE Visualization '05* (2005), pp. 199–206.

[CFM*04] CIGNONI P., FLORIANI L. D., MAGILLO P., PUPPO E., SCOPIGNO R.: Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics 10*, 1 (2004), 29–45.

[CICS05] CALLAHAN S., IKITS M., COMBA J., SILVA C.: Hardware-assisted visibility ordering for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 285–295.

[CMS94] CIGNONI P., MONTANI C., SCOPIGNO R.: Magicsphere: an insight tool for 3d data visualization. *Computer Graphics Forum 13*, 3 (1994), 317–328.

[CMSW04] COOK R., MAX N., SILVA C. T., WILLIAMS P. L.: Image-space visibility ordering for cell projection volume rendering of unstructured data. *IEEE Transactions on Visualization and Computer Graphics 10*, 6 (2004), 695–707.

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003* (2003), pp. 239–248.

[GVW99] GELDER A. V., VERMA V., WILHELMS J.: Volume decimation of irregular tetrahedral grids. *Computer Graphics International* (1999), 222.

[HBH03] HADWIGER M., BERGER C., HAUSER H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003* (2003), p. 40.

[MMFE06] MARROQUIM R., MAXIMO A., FARIAS R., ESPERANCA C.: Gpu-based cell projection for interactive volume rendering. *SIBGRAPI: Brazilian Symposium on Computer Graphics and Image Processing 0* (2006), 147–154.

[NE04] NATARAJAN V., EDELSBRUNNER H.: Simplification of three-dimensional density maps. *IEEE Transactions on Visualization and Computer Graphics 10*, 5 (2004), 587–597.

[PKH04] PIRINGER H., KOSARA R., HAUSER H.: Interactive focus+context visualization with linked 2d/3d scatterplots. In *2nd International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2004)* (July 2004).

[SG98] STAADT O. G., GROSS M. H.: Progressive tetrahedralizations. In *VIS '98: Proceedings of the conference on Visualization '98* (1998), pp. 397–402.

[SS05] SONDERSHAUS R., STRASSER W.: View-dependent tetrahedral meshing and rendering. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (2005), pp. 23–30.

[SS06] SONDERSHAUS R., STRASSER W.: Segment-based tetrahedral meshing and rendering. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (2006), pp. 469–477.

[ST90] SHIRLEY P., TUCHMAN A. A.: Polygonal approximation to direct scalar volume rendering. In *Proceedings San Diego Workshop on Volume Visualization, Computer Graphics* (1990), vol. 24, pp. 63–70.

[THJ99] TROTTS I. J., HAMANN B., JOY K. I.: Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 224–237.

[VFSG06] VIOLA I., FEIXAS M., SBERT M., GRÖLLER M. E.: Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (oct 2006), 933–940.

[WKME03] WEILER M., KRAUS M., MERZ M., ERTL T.: Hardware-based ray casting for tetrahedral meshes. In *Proc. Visualization '03* (2003), pp. 333–340.

[WMFC02] WYLIE B., MORELAND K., FISK L. A., CROSSNO P.: Tetrahedral projection using vertex shaders. In *VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics* (2002), pp. 7–12.

[WZMK05] WANG L., ZHAO Y., MUELLER K., KAUFMAN A.: The magic volume lens: An interactive focus+context technique for volume rendering. *VIS '05: Proceedings of the 16th IEEE Visualization 2005 0* (2005), 47.

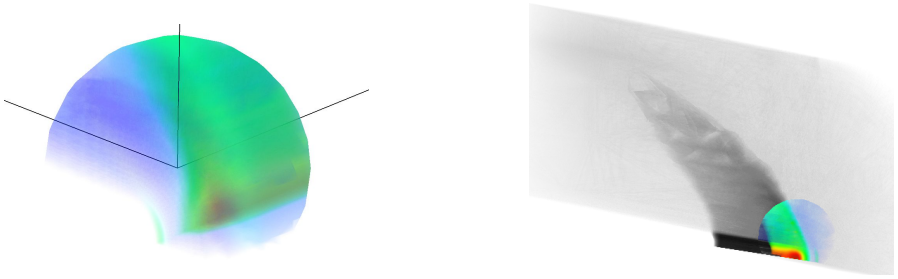


Figure 8: *Rendu Volumique Direct de l'ensemble de données Blunt Fin. La RLI est placée à proximité du choc où la densité de l'air varie fortement (à gauche). Un point de vue différent montre l'intégration du focus au sein du contexte et permet d'obtenir une compréhension globale (à droite).*

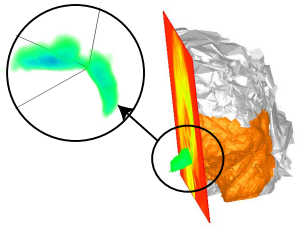


Figure 9: *Integration de plusieurs techniques de rendu pour analyser l'ensemble de données DTI. Un plan de coupe texturé permet de localiser les zones remarquables (en vert). La forme globale du cerveau est affichée via une extraction d'isosurface. Un Rendu Volumique Direct de l'hypothalamus (zoom) et une extraction d'isosurfaces sont réalisés au sein de la RLI.*

Simulation de la Surface de Feuillets β en Modélisation Moléculaire

Loïc Nolin[1,2], Aassif Benassarou[1], Manuel Dauchez[2], Yannick Rémion[1]

[1]Université de Reims Champagne-Ardenne - CReSTIC/LERI
[2]Université de Reims Champagne-Ardenne - UMR CNRS 6198

Abstract
Molecular modeling tools are used both for research, and teaching. Many modes of representation of molecules exist. However, lacks remain on the visualization of very important structures, for example : the β sheets, which confer its stability and function to the molecule. These sheets were never represented as surfaces, meanwhile it would be useful for several reasons. First of all the occasional user could apprehend a sheet much easier by seeing its surface directly. Then, the expert would also have an easier approach of the β sheets, as we can imagine to add informations on the surface which would be useful for him. To complete this work, we used the molecular modeling software BALLView, which is released under GNU/GPL, and modified its sources. We set up a mesh surface based on the structural elements of β sheets. In order to obtain a denser grid leading to a smoother surface, but while preserving the positions of our control points, we used the Catmull-Rom splines, and between each spline we triangulate. Finally we obtain a surface who accounts well for the particular topology of the β sheets.
keywords : *molecular modeling, β sheet, mesh, surface, Catmull-Rom spline.*

Résumé
La modélisation moléculaire est utilisée à la fois pour la recherche, et pour l'enseignement. Il existe de nombreux modes de représentation des molécules. Cependant, des manques subsistent sur la visualisation de structures très importantes, comme par exemple : les feuillets β, qui confèrent à la molécule sa stabilité et sa fonction. Ces feuillets n'ont jamais été représentés sour la forme de surfaces, alors que cela serait utile à plusieurs titres. Tout d'abord l'utilisateur occasionnel pourrait appréhender de façon bien plus aisée un feuillet en voyant directement sa surface. Ensuite, l'expert aurait lui aussi une approche plus facile des feuillets β, car nous pouvons imaginer ajouter des informations sur la surface qui lui seront utiles. Pour réaliser ces travaux nous avons utilisé le logiciel de modélisation moléculaire BALLView, qui est sous licence GNU/GPL, et avons modifié ses sources. Nous avons mis en place un maillage basé sur les éléments structuraux des feuillets β. Afin d'obtenir un maillage plus dense, et une surface moins chaotique, mais tout en conservant les positions de nos points de contrôles, nous nous sommes servis des splines de Catmull-Rom, et entre chaque spline nous avons triangulé. Au final nous obtenons une surface qui rend bien compte de la topologie particulière des feuillets β.
mots clés : *modélisation moléculaire, feuillet β, maillage, surface, spline de Catmull-Rom.*

1. Introduction

La modélisation moléculaire est un outil de recherche précieux aux développements de plus en plus approfondis, notamment dans la recherche de cibles thérapeutiques ou de molécules qui serviront à l'élaboration de nouveaux médicaments.

Il existe différentes façons de représenter une molécule et ses propriétés, le but de ce travail est d'en développer de nouvelles afin d'améliorer encore la compréhension aussi bien du débutant que de l'expert du domaine. Ces représentations auront donc deux buts : le premier est purement pédagogique, il permettra une meilleure appréhension de la

molécule ; et le second est professionnel, le chercheur aura un outil supplémentaire lui permettant d'accélerer ses expériences dans la mesure ou la nouvelle représentation comblera un manque. Il passera donc moins de temps a observer la molécule et a faire « à l'œil » ce que fera le logiciel.

Le travail décrit ici consiste en la représentation des feuillets β (définis dans le paragraphe suivant) d'une molécule sous la forme d'une surface. De cette façon le rendu sera plus clair pour l'utilisateur, et il sera possible d'ajouter des informations propres à chacun des feuillets β directement sur la surface correspondante. En plus de l'aspect qualitatif nous serons capables de quantifier les surfaces de nos feuillets β et ainsi pouvoir suivre leurs évolutions au cours du temps, notamment lors d'une interaction avec une autre molécule.

2. La Modélisation Moléculaire

2.1. Principe

Une molécule est un assemblage d'atomes qui constitue la plus petite partie d'un composé chimique. Une molécule est formée d'au moins deux atomes. Les atomes d'une molécule sont liés entre eux par des liaisons chimiques.

Les macromolécules telles que l'acide désoxyribonucléique (ADN), molécule gigantesque qui contient l'intégralité de notre patrimoine génétique, représentent des milliers, voir des centaines de milliers d'atomes. Une protéine est une macromolécule constituée d'acides aminés. Les acides aminés sont des molécules organiques possédant un squelette carboné (constitué d'atomes de carbone C). Il existe en tout 20 acides aminés. Ils sont les maillons qui constituent les protéines.

Les propriétés des acides aminés (solubilité, propriétés ioniques...) gouvernent la structure de la protéine, que l'on peut décrire à différents niveaux :

La structure primaire : c'est la séquence linéaire des acides aminés dans la protéine, c'est ce que nous appellerons la chaîne dans la suite de cet article. Il n'y a pas de notions géométriques dans la structure primaire. Les liaisons intervenant à ce niveau de structure sont des liaisons très fortes qui se nomment liaisons peptidiques ;

La structure secondaire : elle rend compte de l'organisation de groupes d'acides aminés en éléments structuraux simples, nous parlons ici de géométrie locale. Ces éléments structuraux sont déterminés par la présence de liaisons hydrogènes, ce sont des liaisons faibles en comparaison de la liaison peptidique (environ 20 fois plus faible). Ce sont les hélices α, les feuillets β, les structures en coude et les structures indéterminées. Un feuillet β est constitué de brins β, il est parallèle ou anti-parrallèle suivant que les brins qui le constituent « vont » dans le même sens, où dans le sens opposé. Les liaisons hydrogènes forment des ponts entre deux brins d'un même feuillet. Dans un feuillet anti-parrallèle, deux ponts hydrogènes se forment entre un acide aminé d'un brin et un acide aminé du brin adjacent. Dans un feuillet parallèle, deux ponts hydrogène se forment entre trois acides aminés. Le premier se fait entre un acide aminé *n* d'un brin et un acide aminé *k* du brin adjacent, et le second entre l'acide aminé *n* et l'acide aminé de la chaîne adjacente situé deux résidus plus loin que *k*, donc *k+2* (figure 8). Lorsque plusieurs ponts se suivent, il s'agit d'une échelle. Chaque montant de l'échelle représente un brin β. L'association d'échelles ayant un montant en commun forme un feuillet β ;

La structure tertiaire : elle correspond au repliement de la protéine dans l'espace. Cette structure rend compte de l'organisation entre eux des éléments de structure secondaire. C'est à ce niveau de structure que nous trouvons la géométrie globale de la molécule, nous ne nous en servirons pas dans ces travaux ;

La structure quaternaire : elle définit l'association de protéines. Nous ne nous servirons pas de ce niveau de structure ici.

Les molécules sont invisibles à tout moyen d'investigation direct. La modélisation moléculaire consiste en la construction de modèles tridimensionnels à partir des données de séquences. Elle trouve sa raison d'être d'une part dans les limitations expérimentales des méthodes de détermination de structure de protéines (étude des figures de diffraction des rayons X par un cristal, ou Résonance Magnétique Nucléaire) et d'autre part dans l'incapacité actuelle de prédire la structure 3D à partir de la seule information de séquence. En outre, elle permet d'investiguer les changements de conformations liés à des mutations à partir de structures 3D expérimentales.

C'est pour la première fois en 1958 que Kendrew et ses collaborateurs [KWS*61] ont pu déterminer la structure d'une molécule par cristallographie à rayons X : la myoglobine. Ils ont alors entrepris la construction d'un modèle en laiton à une échelle de 5 cm/Ångstrom. Le modèle fut construit à l'aide de 2500 tiges, à l'intérieur d'un cube de 2 mètres de côté. Des agrafes de couleurs étaient accrochées aux tiges pour représenter les densités électroniques. Malheureusement la véritable forêt que représentaient les tiges, rendit obscure la vision du modèle, et le rendit difficile à ajuster. D'autre part sa taille le rendait très encombrant, et par conséquent, difficile à transporter. C'est en 1966 que Cyrus Levinthal et ses collègues du MIT (Massachusetts Institute of Technology) ont développés un système qui leur permet d'afficher, sur un oscilloscope, en « fil de fer », des représentations de structures moléculaires [Lev66].

Depuis lors l'affichage et les méthodes de rendu des molécules n'ont cessé d'évoluer pour arriver à des modèles de plus en plus complets. Nous nous intéressons plus particulièrement aux méthodes de rendu de la structure secondaire, car ce sont ces structures qui stabilisent la molécule, et ce sont elles qui lui confèrent sa fonction. La méthode de rendu la plus répandue pour représenter la structure secondaire est

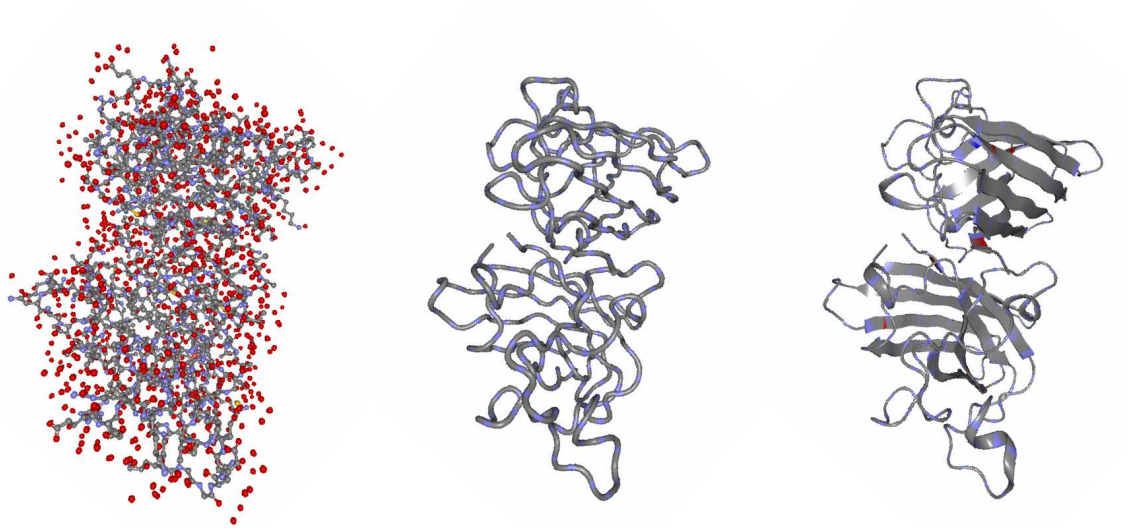


Figure 1: *À gauche le rendu Ball & Stick : les atomes sont des sphères, et les liaisons des cylindres ; au milieu le rendu Backbone (ou squelette) : seul le squelette de la molécule est représenté ; à droite le rendu Cartoon qui représente les structures secondaires : les hélices α sont sous formes de serpentins (en bas à droite sur l'illustration), les brins des feuillets β sont des flèches, les autres structures sont représentées comme le squelette.*

RTyp	Num	Atm	Res	Ch	ResN	X	Y	Z	Occ	Temp	PDB	Line
ATOM	1	N	ASP	L	1	4.060	7.307	5.186	1.00	51.58	1FDL	93
ATOM	2	CA	ASP	L	1	4.042	7.776	6.553	1.00	48.05	1FDL	94

Table 1: *Les deux premières lignes d'un fichier PDB (la ligne commençant par Rtyp n'apparaît pas dans le fichier).*

la méthode « Cartoon ». Les hélices α sont représentées par des « serpentins » ou des cylindres, et les brins β sous formes de flèches (figure 1). Il n'existe aucun mode de rendu qui représente les feuillets dans leur ensemble. C'est le manque que nous proposons de combler par ces travaux.

2.2. La Protein Data Bank (PDB)

En 1971 une base de données concernant la structure tridimensionnelle des protéines a vu le jour dans le Brookhaven National Laboratory (New-York) : la Protein Data Bank (PDB) [BKW*77]. Les scientifiques qui travaillent sur les molécules, sont tenus, s'ils établissent la structure tridimensionnelle d'une molécule inédite, de mettre à jour cette base de données. Ils doivent fournir sous la forme d'un fichier (d'extension .pdb) les coordonnées atomiques de la molécule. Les fichiers PDB sont écrits en clair, il s'agit tout simplement de textes à syntaxe assez simple (table 1).

La première colonne donne le type de donnée, ici il s'agit d'un atome dont le numéro de série se situe dans la deuxième colonne. On trouve ensuite respectivement le type d'élement (C, N, O, S,...), le nom de l'acide aminé (également appelé résidu) auquel il appartient, la chaîne d'acides aminés correspondante, et le numéro du résidu. Suivent les coordonnées spatiales de l'atome, des informations fonctionnelles (facteur d'occupation, de température, ...), identifiant PDB et le numéro de la ligne du fichier.

Les Fichers PDB peuvent contenir beaucoup d'autres informations dont la structure secondaire de la molécule. À chaque acide aminé il est donné une conformation : α, β, coude ou indéterminée. Pour ce travail nous nous interressons à la conformation β (table 2).

La première colonne nous informe sur le type de structure secondaire dont il s'agit, ici un feuillet β (SHEET). Suit le numéro du brin dans le feuillet, puis l'identifiant du feuillet et le nombre de brins dans ce feuillet. Ensuite viennent les identifiants du résidu qui commence le brin (nom, identifiant des chaînes et numéro du résidu dans la séquence). Les colonnes suivantes signifient la même chose, mais pour le résidu terminal du brin. Puis vient le sens du brin, 0 correspond au brin initial, -1 si le brin est anti-paralléle par rapport à son précédent, et 1 s'il est parallèle. Ensuite s'il ne s'agit pas du brin initial on trouve l'identifiant du premier atome du brin (nom, nom et numéro du résidu auquel il appartient). Les colonnes suivantes signifient la même chose mais pour le dernier atome du brin.

RecN	**SNb**	**SI**	**Nb**	**iRN**	**iC**	**iS**	**eRN**	**eS**	**Ss**	**cA**	**cR**	**cRN**	**pA**	**pRN**	**pR**
SHEET	1	S1	5	THR	A	107	ARG	110	0						
SHEET	2	S1	5	ILE	A	96	THR	99	-1	N	LYS	98	O	THR	107
SHEET	3	S1	5	ARG	A	87	SER	91	-1	N	LEU	89	O	TYR	97
SHEET	4	S1	5	TRP	A	71	ASP	75	-1	N	ALA	74	O	ILE	88
SHEET	5	S1	5	GLY	A	52	PHE	56	-1	N	PHE	56	O	TRP	71

Table 2: *Lignes représentants un feuillet* β *dans un fichier PDB (la ligne commençant par RecN n'apparaît pas dans le fichier).*

3. Représentation d'un feuillet β

Avant toute chose, il a été nécessaire de rechercher un logiciel de modélisation moléculaire sur lequel se baser afin de ne pas avoir à recoder ce qui existe déjà. Il existe beaucoup de logiciels de modélisation moléculaire. Parmis les plus connus nous pouvons citer VMD [HDS96], PyMol [Del02], RasMol [SMW95] et MolMol [KBW96]. Notre choix s'est arrêté sur le logiciel BALLView [MHLK05, MHLK06] et ce pour plusieurs raisons. Tout d'abord il reprend tous les éléments de base de la modélisation moléculaire, on y retrouve l'ensemble des représentations les plus classiques : Ball & Stick, Backbone, Cartoon, Van Der Waals, Solvent Excluded/Accessible Surface (figures 1 et 7). Ensuite, il est codé en C++, et son interface est en QT (traditionnellement on retrouve une interface en Python dans ce type de logiciel) qui sont les outils généralement dans le laboratoire. Pour finir, BALLView est sous licence GNU/GPL ce qui nous permet d'utiliser et de modifier ses sources au besoin.

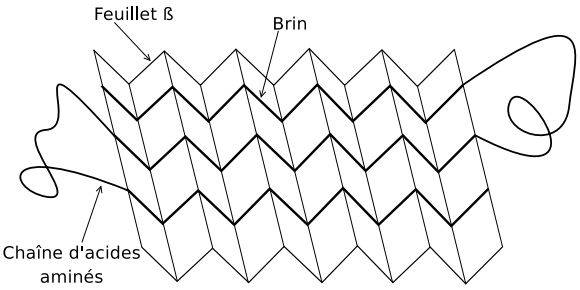


Figure 2: *Représentation d'un feuillet plissé* β.

3.1. Récupération des données nécessaires

Le but de ce travail est de représenter les feuillets β sous la forme de surfaces. Nous venons de voir le type de données dont nous disposons, nous devons récuperer uniquement les données concernant les feuillets β. BALLView peut lire les fichiers PDB, mais il a été nécessaire d'investiguer le code source en profondeur afin d'en comprendre le fonctionnement. Une fois cette étape passée nous avons été en mesure de récupérer uniquement les données concernant les feuillets β. Nous pouvions traiter l'ensemble des atomes dont les résidus sont en conformation β, mais nous ne voulons traiter que ceux qui représentent véritablement le « squelette carboné » de la molécule : ce sont les carbones α. Il n'y a qu'un carbone α par résidu, il est nommé CA dans les fichiers PDB (ligne 2 de la table 1). Nous nous servons du code existant de BALLView pour récuperer ces données. Il est bien évident que BALLView dispose d'un analyseur de fichiers PDB, nous nous sommes donc greffés à cet analyseur, et l'avons modifié afin de ne prendre que ce qui nous intéresse. À savoir dans un premier temps les numéros des résidus en conformation β, le numéro du brin auquel il appartient, l'identifiant du feuillet correspondant au brin, et, il ne faut pas l'oublier : le sens du brin par rapport à son précédent. Après cela, BALLView lit l'ensemble des coordonnées spatiales des atomes pour pouvoir les représenter, à ce moment la dès que le programme traite un des résidu qui nous intéresse nous prenons les coordonnées du carbone α de ce résidu.

3.2. Création du maillage

Nous disposons désormais d'une liste de tous les carbone α dont le résidu est en conformation β, et nous savons pour chacun à quel brin et à quel feuillet il appartient. Il faut maintenant construire la surface des feuillets.

Il existe beaucoup d'algorithmes pour créer des maillages de surfaces. Mais nous sommes face à un problème très spécifique dans la mesure on nous souhaitons passer d'une structure globalement linéique (la chaîne de la structure primaire) à une structure localement surfacique (le feuillet β). C'est pourquoi nous avons eu besoin de développer notre propre algorithme de maillage. Cet algorithme crée un maillage incrémental par couple de brins consécutifs d'un même feuillet. Il est basé sur de simples tests de distance afin de déterminer quel triangle doit être créé, les brins n'étant pas nécessairement les uns faces aux autres, et ne comportant pas toujours le même nombre de résidus. On obtient alors le résultat représenté sur la figure 3.

On constate que cela produit un maillage assez grossier, et une surface très chaotique (difficile à voir sur la figure). Le caractère accidenté de la surface est dû à la gémétrie même des feuillets β dont le nom complet est en réalité feuillet *plissé* β. Les feuillets forment des plis de façon régulière, tous les brins « ondulent » les uns par rapport aux autres (figure 2).

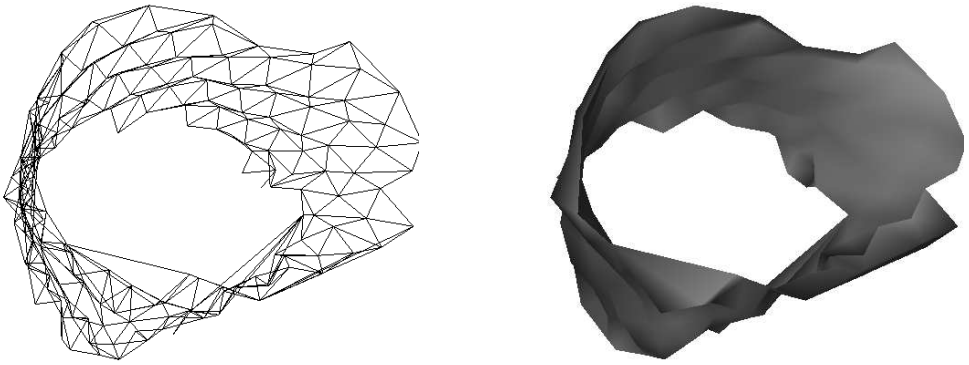


Figure 3: *Feuillet β d'une porine. À gauche : le simple maillage de nos carbones α ; à droite : la surface correspondante.*

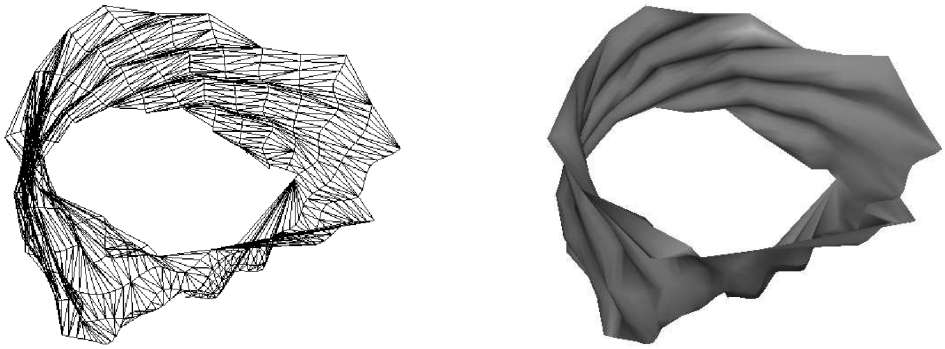


Figure 4: *feuillet β d'une porine. À gauche : le maillage obtenu après calcul des splines de Catmull-Rom ; à droite : la surface correspondante.*

3.3. Raffinement du maillage

Pour obtenir un maillage plus fin il fallait trouver une méthode d'interpolation qui nous permette de conserver les positions de nos points de contrôles, à savoir les positions des carbones α, afin de toujours rester solidaire du squelette de la molécule. Nous avons opté pour les splines de Catmull-Rom [CR74] (figure 5). En effet, ce type de spline interpolante nous permet de bien passer par chacun des points de contrôle, tout en fournissant les ondulations régulières du squelette carboné.

La valeur t, utilisée pour le calcul de la spline, localise le point nouvellement calculé entre deux points de contrôle, sa valeur varie entre 0 et 1. Ces courbes régulières passent par tous les carbones α du brin, à l'exception du premier, et du dernier, puisque les courbes de Catmull-Rom commencent au deuxième point pour finir à l'avant-dernier. Une solution consiste à utiliser les carbones α des résidus précédent et suivant le brin. Si jamais le brin commence ou termine la chaîne, il faut créer un point dans le prolongement du brin. Nous pourrons calculer notre spline de Catmull-Rom avec

$$q(t) = \frac{1}{2} \times \begin{bmatrix} p1 & p2 & p3 & p4 \end{bmatrix} \times \mathcal{M} \times \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\mathcal{M} = \begin{bmatrix} -1 & 2 & -1 & 0 \\ 3 & -5 & 0 & 2 \\ -3 & 4 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Figure 5: *Définition d'une spline de Catmull-Rom.*

l'ensemble de ces points, et ne pas insérer dans le maillage les carbones α supplémentaires.

Le résultat obtenu après raffinement du maillage par les splines de Catmull-Rom est représenté sur la figure 4. On constate que le maillage est beaucoup plus fin, que la surface est beaucoup plus lisse, et beaucoup moins chaotique. Les plissures du feuillet sont apparentes de façon explicite,

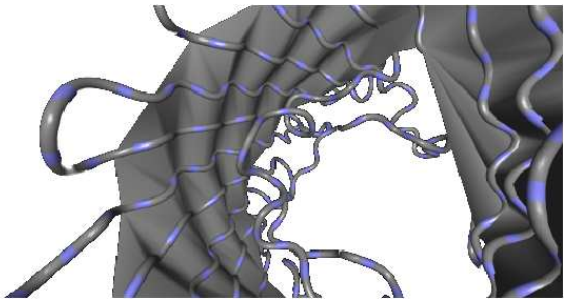


Figure 6: *Représentation d'un feuillet β et du squelette de la molécule.*

ce qui nous donne une information sur le sens du feuillet. Et surtout la surface suit parfaitement le squelette de la protéine. Il est alors possible de quantifier l'aire de son (ou ses) feuillet(s), de faire varier les paramètres de son environnement (la température, la pression...), où de la faire interagir avec une autre molécule (et donc charger un autre fichier PDB), puis de quantifier à nouveau l'aire. C'est une autre application directe de la création de ce genre de surface, la simple visualisation étant la première.

La figure 6 contient la représentation de la surface d'un feuillet, ainsi qu'une autre représentation qui se nomme « backbone » qui permet de visualiser le squelette d'une molécule. C'est une représentation classique que l'on peut observer dans tous les logiciels de modélisation moléculaire. On se rend bien compte sur cette figure que notre surface suit parfaitement la représentation « backbone » de la molécule. Les ondulations des brins sont parfaitements jointives avec le maillage.

4. Cas particuliers

Il arrive que certains fichiers PDB soient incomplets concernant tout ou partie de la structure secondaire. En règle générale soit les informations sur les structures secondaires sont vides, soit il manque l'orientation des brins (table 2 colonne **Ss**).

Lorsque qu'il manque l'orientation des brins, la colonne **Ss** est remplie de 0. Ne sachant pas dans quel sens les parcourir, l'algorithme de maillage renvoie un résultat aberrant. Il faut alors construire un vecteur représentatif de la direction générale et du sens de chaque brin, afin de faire un simple produit scalaire. Aucun brin n'étant, en feuillet β, replié sur lui même, on construit les vecteurs entre les premiers et les derniers résidus du brin. On réalise notre produit scalaire, et cela suffit a déterminer le sens relatif des brins. Si le résultat est positif alors le brin est parallèle par rapport à son précédent, sinon il est anti-parallèle. Cette solution a l'avantage d'être peu coûteuse et efficace. Le résultat peut être observé sur la partie gauche de la figure 9.

Le problème se complique lorsque nous n'avons aucune information sur la structure secondaire d'une molécule, il faut alors utiliser un algorithme de prédiction de structure secondaire, ces algorithmes se basent sur la position des résidus les uns par rapport aux autres. Ce sujet ayant été largement étudié nous disposons de plusieurs algorithmes très efficaces dans ce domaine. Le plus utilisé est celui de Kabsch et Sander [KS83], c'est celui qui est implémenté dans BALLView. Malheureusement l'implémentation ne renvoie pas d'information sur un feuillet ou un brin, elle renvoie seulement si un résidu est en conformation β ou non. Nous ne pouvons donc distinguer ni les différents brins, ni les différents feuillets. Nous devons alors nous mettre à la recherche de ponts, puis d'échelles (section 2.1) afin d'obtenir nos brins et nos feuillets β.

En même temps que l'algorithme de prédiction de structure secondaire, il y a une recherche des ponts hydrogènes. Nous utilisons tous les ponts hydrogènes qui forment un pont entre deux résidus. Nous vérifions quand un pont se fait entre un résidu *n* et un résidu *k*, que le pont suivant se fait entre le résidu *n* et le résidu *k+2*. Dans ce cas, nous sommes en présence de deux brins β parallèles. Et lorsque le pont suivant se fait entre le résidu *n* et le résidu *k*, alors nous sommes en présence de deux brins β anti-paralléles. Nous disposons désormais d'une liste de nos ponts hydrogènes qui correspondent à des résidus en conformation β, il nous faut maintenant trouver les brins et les feuillets. Pour cela, nous recherchons des échelles : une succession de ponts du même type (paralléle ou anti-parallèle). Les échelles qui partagent un montant commun, forment un feuillet β. Le résultat est visible sur la partie droite de la figure 9.

5. Conclusions et Perspectives

La majeure partie de ce travail a été consacrée à l'exploration et à la compréhension du code source de BALLView. Nous nous sommes penchés tout particulièrement sur les parties qui traitent du chargement des fichiers, et sur celles qui traitent des modes de rendu existants.

Le but de ce travail était de pouvoir visualiser la surface d'un feuillet β, nous avons développé un modèle fonctionnel qui reste à améliorer, dans la mesure ou le maillage est perfectible. En effet, sur certaines molécules (rares), un brin β va former une dépression ce qui aura pour effet de déformer un peu le maillage. Cependant les premiers résultats observés sont encourageants et une fois finalisé, ce modèle devrait être un outil performant, aussi bien pour la recherche que pour l'enseignement.

Les perspectives sur ce travail sont multiples maintenant que nous disposons d'une surface cohérente. Nous pouvons faire de la visualisation sémantique en utilisant les outils traditionnels de la synthèse d'image (le bump-mapping, le texte, la couleur, l'opacité...) afin d'y localiser visuellement des données biochimiques dont nous avons besoin. Il faudra aussi travailler sur la symbolique des informations que l'on

souhaite représenter, afin que l'utilisateur puisse, de façon intuitive, comprendre ce qui est affiché.

Une autre perspective serait de pouvoir étendre le feuillet dans le sens, et dans la proportion, que l'on souhaite. Cette fonctionnalité servirait à ce que l'utlisateur se rende compte de l'influence du feuillet sur la molécule, son organisation spatiale, sa fonction.

Enfin, la suite logique de ce travail serait d'adapter ce modèle aux hélices α.

References

[BKW*77] BERNSTEIN F. C., KOETZLE T. F., WILLIAMS G. J., MEYER E. F., BRICE M. D., RODGERS J. R., KENNARD O., SHIMANOUCHI T., TASUMI M. : The protein data bank : a computer-based archival file for macromolecular structures. *Journal of Molecular Biology 112*, 3 (May 1977), 535–542.

[CR74] CATMULL E., ROM R. : A class of local interpolating splines. In *International Conference on Computer Aided Geometric Design '74* (1974), pp. 317–326.

[Del02] DELANO W. L. : The pymol molecular graphics system, 2002.

[HDS96] HUMPHREY W., DALKE A., SCHULTEN K. : VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics 14* (1996), 33–38.

[KBW96] KORADI R., BILLETER M., WÜTHRICH K. : Molmol : a program for display and analysis of macromolecular structures. *J Mol Graph 14*, 1 (1996), 51–5, 29–32.

[KS83] KABSCH W., SANDER C. : Dictionary of protein secondary structure : pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers 22*, 12 (Dec. 1983), 2577–2637.

[KWS*61] KENDREW J., WATSON H., STRANDBERG B., DICKERSON R., PHILLIPS D., SHORE V. : The amino-acid sequence x-ray methods, and its correlation with chemical data. *Nature 190* (1961).

[Lev66] LEVINTHAL C. : Molecular model-building by computer. *Scientific American 214*, 6 (1966), 42–52.

[MHLK05] MOLL A., HILDEBRANDT A., LENHOF H.-P., KOHLBACHER O. : Ballview : An object-oriented molecular visualization and modeling framework. *Journal of Computer-Aided Molecular Design 19*, 11 (Nov. 2005), 791–800.

[MHLK06] MOLL A., HILDEBRANDT A., LENHOF H.-P., KOHLBACHER O. : Ballview : a tool for research and education in molecular modeling. *Bioinformatics 22*, 3 (2006), 365–366.

[SMW95] SAYLE R. A., MILNER-WHITE E. J. : Rasmol : biomolecular graphics for all. *Trends in Biochemal Sciences 20*, 9 (September 1995).

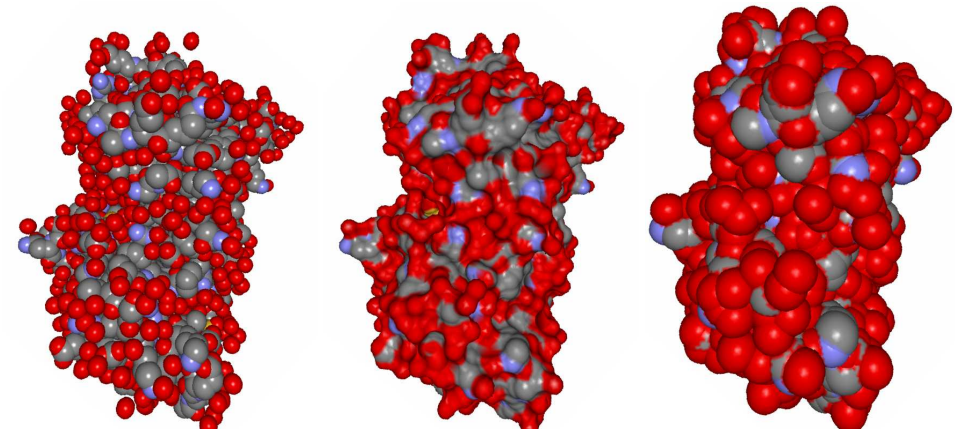


Figure 7: *À gauche le rendu Van Der Waals représente les forces du même nom ; au milieu le rendu Solvent Excluded Surface représente la surface de la molécule qui n'est pas accessible au solvant ; à droite le rendu Solvent Accessible Surface représente la surface accessible au solvant.*

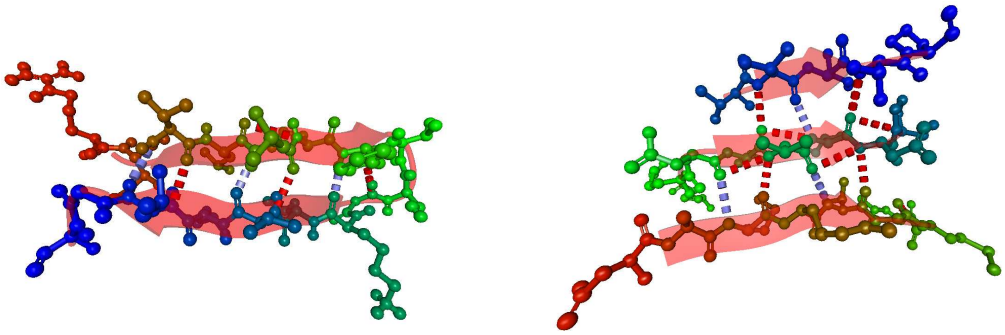


Figure 8: *Représentation de brins β anti-parallèles à gauche, et parallèles à droite. Les pointillés représentent les ponts hydrogènes, les flèches, le sens des brins.*

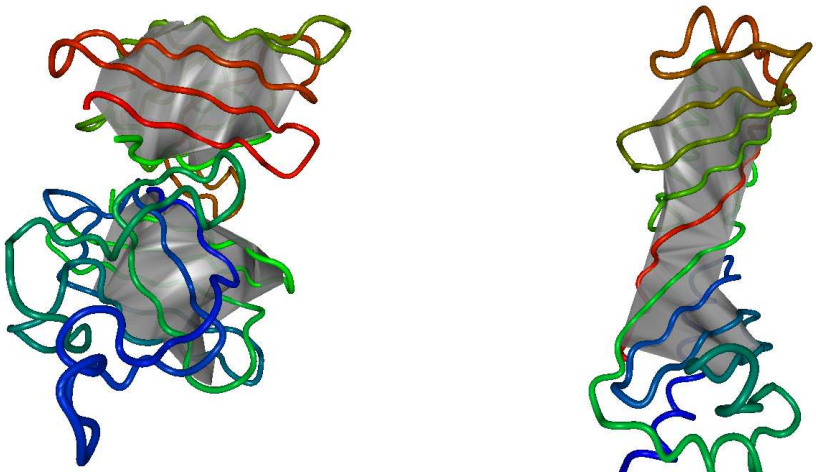


Figure 9: *À gauche la représentation d'un feuillet dont le fichier PDB ne contenait pas d'informations sur le sens des brins ; à droite la représentation d'un feuillet dont le fichier PDB ne contenait aucune informations sur la structure secondaire de la molécule.*

Un modèle générique pour la manipulation de maillages multirésolution

Pierre Kraemer [1], David Cazier [1], Dominique Bechmann [1]

[1]LSIIT, UMR 7005 CNRS Université de Strasbourg

Résumé

Les cartes multirésolution et leur structure de données associée ont été introduites dans [KCB07b, KCB07a]. En tant qu'extension des cartes combinatoires, ce modèle hérite naturellement de sa généralité, de sa robustesse et de l'efficacité des structures de données qui en sont dérivées.
Nous exploitons ici la généralité de ce modèle dans deux approches différentes. D'une part nous l'utilisons pour représenter et manipuler des surfaces de subdivision multirésolution générées par des schémas qui ne sont généralement pas supportés par les structures classiques. Ceci est illustré avec des outils d'édition multirésolution construits sur les schémas de subdivision quad/triangle et $\sqrt{3}$. D'autre part, nous l'utilisons dans une approche de décimation pour construire une hiérarchie de maillages à partir d'une triangulation quelconque. Dans ce contexte, la capacité à parcourir les niveaux intermédiaires directement et sans avoir à reconstruire la topologie permet d'améliorer l'efficacité des algorithmes multirésolution appliqués sur ces maillages.

1. Introduction

Les maillages multirésolution sont au coeur de nombreux travaux et applications en informatique graphique. Plusieurs méthodes existent pour obtenir de tels maillages. Les modèles et structures utilisés pour leur représentation et leur manipulation sont souvent spécifiques à l'application visée et souffrent alors d'une certaine rigidité.

Notre objectif est de définir un modèle suffisamment souple et efficace pour être utilisé dans différentes approches manipulant des maillages multirésolution. Nous illustrons cela dans le cadre des surfaces de subdivision multirésolution et des maillages progressifs.

1.1. Surfaces de subdivision multirésolution

La plupart des travaux sur les surfaces de subdivision multirésolution sont basés sur des schémas primaux utilisant la quadrisection de face en tant qu'opération topologique de raffinement de base. Généralement, les structures de données utilisées sont déduites des règles de subdivision : arbres quaternaires de triangles pour les schémas utilisant la quadrisection de triangles, arbres quaternaires de carrés pour ceux opérant une quadrisection de carrés. En plus de problèmes d'efficacité et de consistance topologique déjà soulevés, ces structures sont limitées à la représentation des maillages pour lesquels elles ont été conçues. D'autres schémas de subdivision peuvent être utilisés dans le cadre de la multirésolution et y apporter leurs spécificités.

Le schéma quad/triangle [SL03] propose d'unifier les schémas de subdivision primaux triangulaires – tel que Loop [Loo87] – et carrés – tel que Catmull-Clark [CC78]. Les auteurs de ce schéma partent de deux observations. Tout d'abord, de nombreux maillages conçus manuellement sont composés de zones plus naturellement carrées ainsi que de zones plus naturellement triangulaires. Deuxièmement, l'utilisation d'un schéma triangulaire sur un maillage carré triangulé, tout comme celle d'un schéma carré sur un maillage triangulaire, aboutit à des surfaces limites de moins bonne qualité.

Le schéma $\sqrt{3}$ [Kob00] travaille sur des maillages triangulaires. Il a un certain nombre d'avantages par rapport aux autres schémas triangulaires tels que Loop [Loo87] ou Butterfly [DLG90] : moins de triangles sont générés à chaque pas de subdivision (le nombre de faces n'est multiplié que par 3); les masques utilisés pour la calcul de la géométrie sont compacts; la subdivision adaptative évite complètement

les problèmes liés à l'apparition de trous topologiques dans le maillage.

Pour représenter des maillages multirésolution générés par de tels schémas une structure suffisamment générale est nécessaire. Les cartes multirésolution sont un exemple d'une telle structure. Ceci est illustré avec des outils d'édition multirésolution basés sur ces schémas.

1.2. Décimation

De nombreux maillages sont aujourd'hui issus de processus d'acquisition. Ceux-ci sont généralement très denses et ne présentent pas de régularité particulière. De nombreux travaux ont été menés afin de pouvoir appliquer à ces objets les algorithmes issus des études sur les maillages multirésolution. Une première approche consiste construire une surface de subdivision multirésolution approchant au mieux l'objet original en partant d'une version simplifiée du maillage d'origine [LSS*98]. La connectivité de départ est dans ce cas perdue.

D'autres approches ont été développées afin de conserver la connectivité d'origine. Celles-ci sont inspirées des maillages progressifs introduits dans [Hop96]. Les applications vont de l'édition multirésolution au filtrage [GSS99] en passant par la compression de maillage [PR00, Ber07].

Un maillage progressif est construit en opérant une succession de contractions d'arêtes dans le maillage. Ces dernières peuvent ensuite être inversées en appliquant les éclatements de sommets correspondants. Les structures de données utilisées pour coder les maillages progressifs sont généralement conçues de la façon suivante. Le maillage le plus grossier est stocké ainsi qu'un ensemble d'opérations d'éclatement de sommet. Chacun de ces noeuds doit permettre d'identifier le lieu de l'éclatement et contenir l'information nécessaire à la reconstruction. Cet ensemble peut être une simple file inversant l'ordre des contractions d'arête, ou organisé en un arbre de dépendances permettant une reconstruction adaptative du maillage. D'une manière générale, cette structure peut être vue comme un arbre binaire de sommets.

Tout comme les algorithmes de subdivision, ceux développés dans le cadre des maillages progressifs font grand usage des opérateurs topologiques de voisinage à des fins de calculs géométriques. La topologie des maillages intermédiaires doit donc être disponible afin de faire circuler l'information dans la hiérarchie. Les structures basées sur des arbres nécessitent la reconstruction des maillages intermédiaires pour retrouver cette information. Nous montrons ici comment les cartes multirésolution permettent d'améliorer l'efficacité des algorithmes appliqués à ces maillages en fournissant des parcours topologiques efficaces de tous les maillages intermédiaires.

1.3. Plan

Nous commençons cet article avec un rappel sur la structure des cartes multirésolution. Nous montrons ensuite comment elle peut être appliquée à la représentation de surfaces de subdivision multirésolution générées avec les schémas quad/triangle et $\sqrt{3}$. La section suivante expose l'utilisation de cette structure dans le cadre des maillages progressifs. Enfin nous concluons et exposons quelques perspectives à ce travail.

2. Cartes multirésolution

Nous faisons ici une présentation informelle des cartes multirésolution de dimension 2 en termes de structure de données. Celle-ci peut alors être vue comme une extension de la structure de demi-arêtes. Nous faisons d'abord quelques rappels rapides sur cette dernière, puis nous présentons son extension multirésolution.

2.1. Demi-arêtes

La structure de demi-arêtes utilise les adjacences entre les arêtes pour représenter la topologie du maillage de 2-variétés orientables. Chaque arêtes est constituée de deux demi-arêtes symétriques. Deux définitions peuvent être données appelées primale (figure 1a) et duale (figure 1b). Dans les deux cas, chaque demi-arête dispose d'un lien vers sa demi-arête *opposée*, *suivante* et *précédente*.

(a) Primal (b) Dual

Figure 1: *La structure de demi-arêtes*

Plonger une telle structure consiste à associer des données aux entités topologiques ou cellules du maillage. Par exemple un point 3D est associé à chaque sommet du maillage. Dans ce cas, chaque demi-arête a un lien vers un point 3D, et toutes les demi-arêtes associées au même sommet partagent un même lien.

2.2. Extension multirésolution

2.2.1. Présentation

Une structure de demi-arêtes multirésolution est une hiérarchie de structures de demi-arêtes. Comme illustré à la figure 2, les demi-arêtes sont réparties dans des ensembles distincts correspondants aux niveaux de résolution auxquels elles ont été insérées. Les demi-arêtes du maillage de départ (niveau 0) appartiennent à l'ensemble L^0. A chaque niveau de résolution suivant i, un ensemble de demi-arêtes L^i est introduit et ajouté aux demi-arêtes existantes.

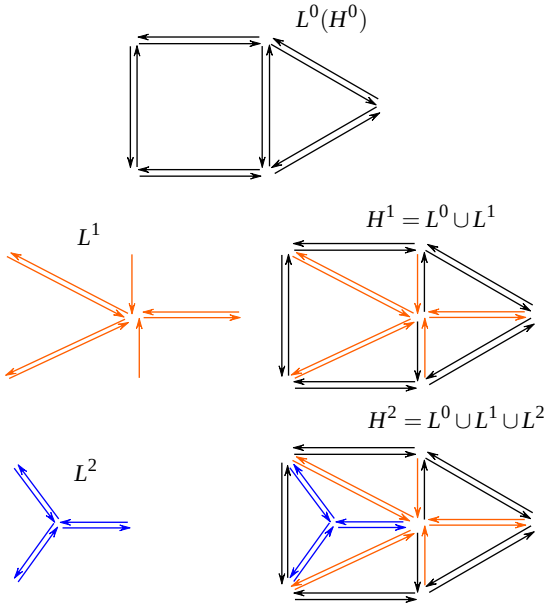


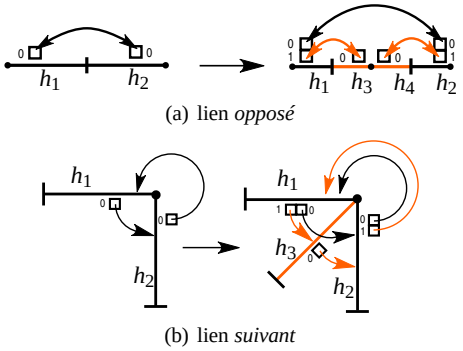
Figure 2: *Ensembles de demi-arêtes*

Soit H^i l'ensemble de demi-arêtes insérées aux niveaux inférieurs ou égaux à i. On a $H^i = \bigcup_{j=0}^{i} L^j$. Ces ensembles forment une suite $\{H^i\}_{i\geq 0}$ d'ensembles telle que : $H^0 \subset H^1 \subset H^2 \subset \ldots \subset H^i \subset \ldots$. Il n'y a pas plus de demi-arêtes dans la structure que le nombre nécessaire à la description du maillage le plus fin.

Une demi-arête est présente dans le maillage à tous les niveaux suivants son niveau d'insertion. Ses liens topologiques ne restent généralement pas égaux d'un niveau à l'autre. Ceux-ci doivent donc être indexés par le niveau. En d'autres termes, chaque demi-arête dispose pour chaque type de relation (*opposé*, *suivant*), d'un ensemble de liens correspondant aux différents niveaux de résolution. Soit a le niveau d'insertion d'une demi-arête et k le niveau de résolution maximum du maillage. Etant donné qu'une demi-arête ne peut avoir de lien à un niveau précédent son niveau d'insertion, elle n'a besoin de disposer que de $(k-a)$ liens. Si ceux-ci sont stockés dans un tableau, on retrouve le lien de niveau i dans la case d'indice $(i-a)$ de ce tableau.

Les liens vers le plongement sont aussi indexés de la sorte par le niveau de résolution. Chaque demi-arête dispose d'un ensemble de liens vers des points 3D, correspondants aux positions des sommets associés aux différents niveaux. Cela est géré exactement de la même manière que les liens topologiques.

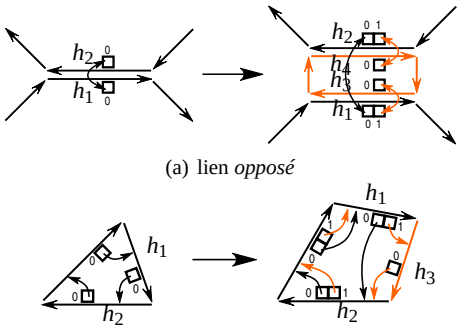
Le maillage correspondant à chaque niveau i intermédiaire peut être parcouru immédiatement et aussi efficacement qu'une simple structure de demi-arêtes. Cela est réalisé en



(a) lien *opposé*

(b) lien *suivant*

Figure 3: *Indexation des liens en version primale*



(a) lien *opposé*

(b) lien *suivant*

Figure 4: *Indexation des liens en version duale*

considérant les demi-arêtes de l'ensemble H^i ainsi que leurs relations et plongements de niveau i. Ainsi, à chaque niveau de résolution, les requêtes d'adjacence sont exécutées de manière optimale.

2.2.2. Adaptativité

Dans le contexte de la subdivision, l'adaptativité signifie que la profondeur de subdivision peut varier dans le maillage suivant un critère donné (courbure locale, distance à la surface limite, point de vue, ...). Chaque pas de subdivision n'est donc pas appliqué de manière régulière sur tout le maillage. Le but est ici d'économiser la mémoire qui aurait été dépensée en subdivisant entièrement le maillage.

Suivant un certain critère, il est décidé que certaines zones du maillage n'ont plus à être subdivisées à partir d'un niveau de résolution l donné. Cela implique que pour certaines demi-arêtes de H^l, les relations de niveau $l+1$ seront différentes, alors que pour d'autres, elles seront égales aux relations de niveau l. Dupliquer les liens identiques ne serait pas efficace en termes d'occupation mémoire.

Cela est donc géré de la manière suivante : si un lien ne change pas entre deux niveaux de résolution, alors rien n'est ajouté au tableau de relations correspondant. La manière

d'accéder aux liens est donc changée en conséquence : si rien n'est défini au niveau requis (c'est-à-dire si le tableau est trop petit), on accède au lien défini au niveau le plus fin existant. Le même traitement est appliqué pour gérer l'accès au plongement.

Dans un objet subdivisé de manière adaptative, les maillages obtenus sont composés de cellules de niveaux différents. Le maillage de niveau i est composé de cellules de niveau inférieur ou égal à i. L'information de profondeur est nécessaire pour pouvoir continuer et mettre à jour la subdivision. Nous allons voir que les données déjà contenues dans les demi-arêtes suffisent à déduire cette information.

3. Le schéma quad/triangle

Comme dans tout schéma de subdivision, deux étapes peuvent être distinguées dans le schéma quad/triangle : raffiner la topologie et calculer la géométrie. Le raffinement de la topologie consiste ici en de la quadrisection de face, en conservant des triangles dans les zones triangulaires, et des carrés dans le reste du maillage. Cela ne cause pas de problème topologique aux frontières entre les zones triangulaires et carrées car dans les deux cas ce raffinement consiste à couper les arêtes et en insérer de nouvelles. Différentes stratégies peuvent être appliquées pour calculer la géométrie, et plus de détails peuvent être trouvés dans [SL03, PS04, SW05].

La figure 5 illustre un détail de maillage aux niveaux de résolution 0 et 1. Les tableaux de liens *opposé* sont illustrés pour deux arêtes. Les demi-arêtes en gras sont celles appartenant à L^1. Cette opération de raffinement topologique est appliquée exactement comme elle le serait sur un maillage mono-résolution, à l'exception que les nouvelles demi-arêtes sont insérées et liées aux existantes dans le nouveau niveau de résolution. Ceci permet de conserver l'ancien niveau et de pouvoir continuer à y accéder.

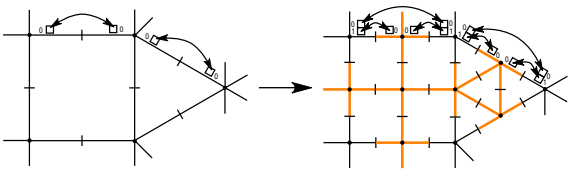


Figure 5: *Quad/triangle avec des demi-arêtes multirésolution*

La valence des sommets n'est pas modifiée par une telle opération de subdivision. Les liens *suivant* des demi-arêtes ne changent donc jamais d'un niveau de résolution à l'autre. La manière de gérer l'adaptativité décrite ci-dessus évite de stocker cette information redondante, et un seul lien *suivant* est donc stocké pour chaque demi-arête.

Comme nous l'avons vu plus haut, des cellules correspondant à des profondeurs de résolution différentes coexistent dans un maillage généré de manière adaptative (tel que celui illustré figure 6). Cette information doit être connue afin de pouvoir continuer à mettre à jour le maillage correctement. Sans aucun coût de stockage supplémentaire, on peut aisément déterminer la profondeur d'une arête. Celle-ci est égale au maximum des niveaux d'insertion des deux demi-arêtes qui la composent. A partir de cette information, on peut déduire le niveau d'une face comme le minimum des niveaux de ses arêtes incidentes, et le niveau d'un sommet comme le maximum des niveaux de ses arêtes incidentes.

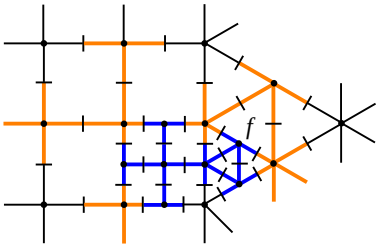


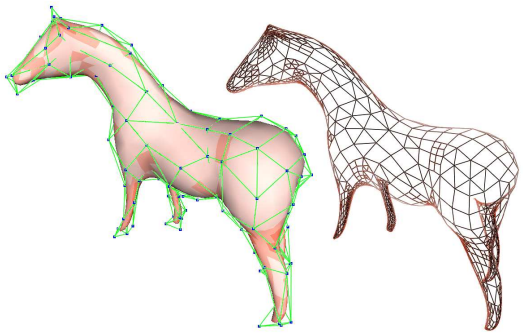
Figure 6: *Maillage quad/triangle adaptatif*

Comme c'était le cas dans la subdivision adaptative de maillages entièrement triangulaires ou carrés, les demi-arêtes multirésolution permettent de conserver la consistance topologique du maillage en évitant la création de trous entre les faces de niveaux différents. Cela est dû à la gestion des arêtes dans le processus de subdivision et à la capacité du modèle à représenter des faces polygonales quelconques.

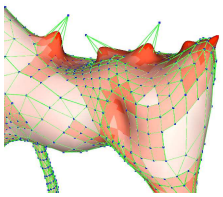
Un effet supplémentaire à gérer ici est illustré avec la face f du maillage de la figure 6. Au niveau de résolution 1, cette face était un triangle. Après un pas de subdivision adaptative, elle devient un quadrilatère au niveau 2. Pour être capable d'appliquer l'opérateur de raffinement adéquat sur cette face quand elle viendra à être subdivisée, il faut pouvoir retrouver que cette face était à l'origine un triangle.

Cette information peut être déduite simplement et à nouveau sans aucun coût mémoire supplémentaire. Soit l_f le niveau d'une face dans le maillage de niveau l (dans notre exemple, f est une face de niveau 1 dans un maillage de niveau 2); le nombre d'arêtes originel d'une face est son nombre d'arêtes au sein du maillage de niveau l_f.

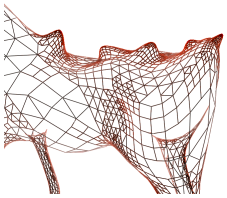
La figure 7 montre un exemple d'objet obtenu avec un outil d'édition multirésolution basé sur la subdivision quad/triangle (la couleur est fonction de la profondeur de subdivision). L'adaptivité est ici dirigée par un critère de courbure locale. Le maillage est mis-à-jour dynamiquement durant l'édition (subdivision et simplification) afin de toujours satisfaire ce critère. Une restriction non-imposée par la structure est ajoutée ici, interdisant à deux faces voisines d'avoir plus d'un niveau de différence. Cela permet, lors de l'application des masques de subdivision, d'avoir toujours à disposition l'ensemble des sommets nécessaires au calcul d'une position.



(a) Un maillage quad/triangle



(b) Après quelques éditions



(c) Après quelques éditions (fil de fer)

Figure 7: *Example d'objet*

4. Le schéma $\sqrt{3}$

Le schéma $\sqrt{3}$ travaille sur des maillages triangulaires. L'opération de raffinement topologique consiste en la procédure suivante : une division en 3 de chaque triangle est effectuée en insérant un nouveau sommet en son centre; ceci introduit trois nouvelles arêtes connectant le nouveau sommet central aux anciens sommets du triangle; les arêtes originelles sont basculées pour régulariser la valence des sommets.

Ce schéma peut également être appliqué de manière adaptative. Si un triangle est subdivisé alors que son voisin ne l'est pas, leur arête commune n'est pas basculée. Deux types de triangles différents, appelés *pair* et *impair*, doivent alors être considérés. Les triangles dits *impairs* sont ceux obtenus par la division en 3 d'un triangle quand le basculement d'arête n'a pas eu lieu. Ces triangles forment la frontière entre les zones ayant des profondeurs de subdivision différentes.

Dans les figures suivantes la subdivision d'un détail de maillage triangulaire composé de deux faces est illustrée. Les relations topologiques *suivant* sont dessinées pour chaque demi-arête. La relation *opposé* n'est quant à elle jamais modifiée durant un pas de subdivision. A l'instar de la relation *suivant* dans le cas de la subdivision quad/triangle, un seul lien est donc stocké par demi-arête pour cette relation.

La figure 8 montre le maillage au niveau de résolution

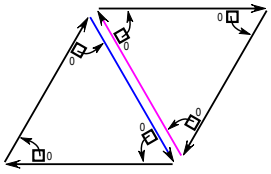
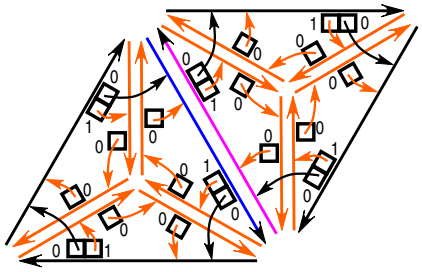
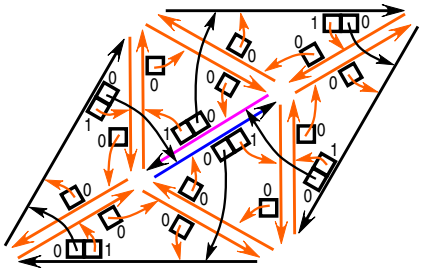


Figure 8: *Deux triangles au niveau 0*



(a) Niveau 1 après la division en 3



(b) Niveau 1 après le basculement d'arête

Figure 9: *Subdivision* $\sqrt{3}$

0. Chaque triangle est composé de trois demi-arêtes ayant chacune une relation *suivant*. Les deux demi-arêtes formant l'arête commune sont colorées ici en bleu et mauve afin de pouvoir les distinguer par la suite après le basculement d'arête.

La figure 9a montre le maillage au niveau de résolution 1 après la division en 3 des triangles. Pour chaque face de départ, trois arêtes (six demi-arêtes) sont insérées au nouveau niveau. Les demi-arêtes de l'ensemble L^0 ont maintenant chacune deux liens *suivant*.

Dans la figure 9b, le basculement d'arête a été effectué dans le sens anti-horaire. Cette opération est exécutée **au sein** du maillage de niveau 1, c'est-à-dire que les relations de niveau 1 ont été altérées. Les relations de niveau 0 des demi-arêtes de l'arête basculée sont toujours disponibles et les faces du maillage de niveau 0 peuvent toujours être parcourue normalement.

Il faut faire attention ici à ne pas être être trompé par la représentation graphique. Ces figures illustrent uniquement la manipulation de l'information topologique et ne de-

vraient pas être interprétée géométriquement. L'information de plongement attachée aux demi-arêtes est gérée tout comme les relations topologiques et est également indexée par le niveau de résolution. Ainsi le plongement de niveau 0 associé aux demi-arêtes de L^0 n'est pas perdu lors d'un pas de subdivision, même après le basculement d'arête.

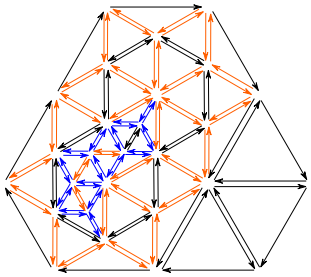


Figure 10: *Niveau 2 après une subdivision adaptative*

Cette opération de base peut être appliquée de manière régulière sur un maillage triangulaire le nombre souhaité de fois. Une subdivision adaptative peut également être réalisée : la figure 10 illustre un exemple de maillage sur lequel deux pas de subdivision adaptative ont été appliqués. Les relations entre les demi-arêtes ne sont plus dessinées ici. Les demi-arêtes oranges appartiennent à l'ensemble L^1 et les bleues à l'ensemble L^2.

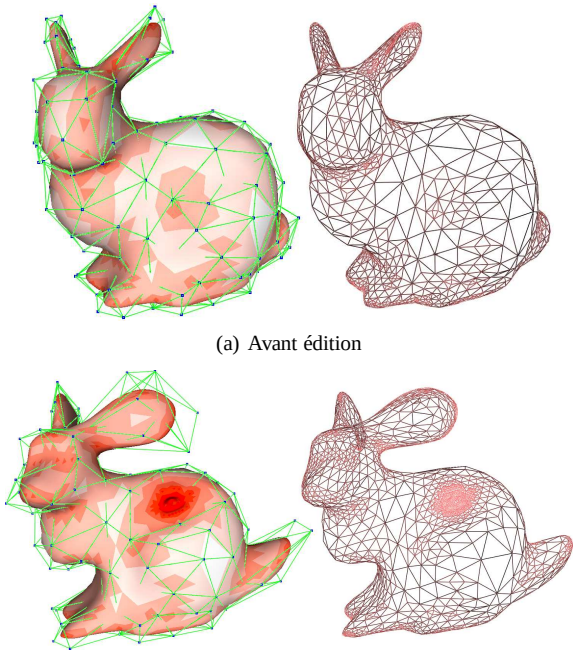
Comme précédemment, des faces appartenant à des profondeurs de subdivision différentes coexistent dans un maillage généré par subdivision adaptative, et cette information est cruciale pour une mise-à-jour correcte du maillage. A nouveau, aucune donnée supplémentaire n'est nécessaire pour récupérer le niveau d'une face. Celui-ci est égal au maximum des niveaux d'insertion de ses demi-arêtes.

Caractériser la parité d'une face est tout aussi important pour continuer la subdivision. Cette information peut être récupérée de la façon suivante : une face *paire* est entourée par des faces de son propre niveau ou supérieur; une face *impaire* est entourée par des faces de son propre niveau ou inférieur (des cas particuliers doivent être traités pour les bords).

Un exemple de maillage obtenu dans un outil d'édition multirésolution basé sur la subdivision $\sqrt{3}$ est illutré à la figure 11. L'adaptivité est dirigée ici par un critère de courbure locale et le maillage est également mis-à-jour dynamiquement durant l'édition.

5. Décimation

La structure de demi-arête multirésolution peut également être utilisée dans une autre approche, en construisant la hiérarchie non plus en subdivisant successivement un maillage de base, mais en simplifiant au fur et à mesure un maillage fin donné. Nous rejoignons ici le principe des pyramides



(a) Avant édition

(b) Après quelques éditions

Figure 11: *Exemple d'objet*

de cartes de [BK03,GSDL06]. Ces dernières sont construites en utilisant des opérateurs de contraction et suppression de cellules. Les applications visées se situent dans le domaine de l'analyse et la segmentation d'images 2D et 3D.

Nous appliquons notre modèle à la construction de maillages multirésolution en utilisant des techniques de décimation basées sur les opérateurs de contraction d'arête et son inverse, l'éclatement de sommet, illustrés à la figure 12. La simplification de maillages basée sur ces opérateurs a été rendue populaire avec l'introduction des maillages progressifs [Hop96]. Ceux-ci ont été développés en visant des applications telles que le stockage et la transmission progressive de gros maillages triangulaires. Cette technique permet également l'accès à différents niveaux d'approximation du maillage.

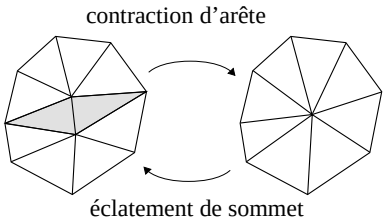


Figure 12: *Contraction d'arête et éclatement de sommet*

Des méthodes visant à traiter la géométrie ont ensuite vu le jour en se basant sur la hiérarchie de maillages

obtenue lors de la construction d'un maillage progressif. Celles-ci visent par exemple à la compression et décompression progressive de maillages triangulaires quelconques [PR00, Ber07], ou à permettre l'édition multirésolution et le filtrage sur des triangulations non-régulières [GSS99].

Ces algorithmes font, tout comme les algorithmes de subdivision, grand usage de l'information topologique du maillage afin d'accéder au voisinage des sommets. Un modèle fournissant des requêtes d'adjacences optimales à tous les niveaux peut donc apporter un gain d'efficacité important à l'exécution de ces algorithmes.

La figure 13 illustre l'opération topologique effectuée lors d'une contraction d'arête effectuée sur le maillage de niveau 0 dans la structure des demi-arêtes multirésolution. Le maillage de gauche est le maillage de niveau 0 avant simplification. Il est composé des demi-arêtes appartenant à l'ensemble L^0. Une contraction d'arête supprime deux faces triangulaires du maillage. Les demi-arêtes correspondant à ces deux triangles sont alors ajoutées à l'ensemble des demi-arêtes du niveau suivant (plus fin), c'est-à-dire ici L^1. Les triangles adjacents sont ensuite cousus entre eux au niveau 0 pour former le nouveau maillage de niveau 0 simplifié.

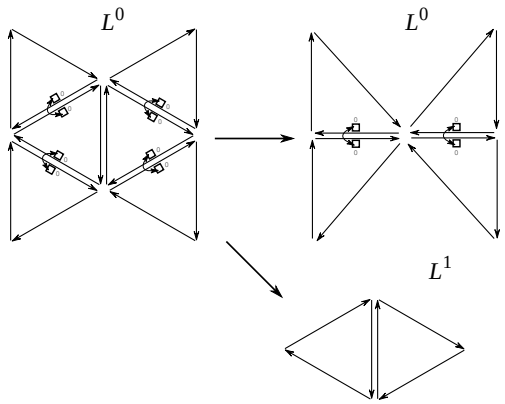


Figure 13: *Contraction d'arête*

Tel qu'illustré à la figure 14 les demi-arêtes nouvellement cousues au niveau 0 ont maintenant un lien *opposé* supplémentaire. Leur ancienne relation devient celle de niveau 1, tandis que la nouvelle est celle de niveau 0.

Une passe de simplification peut consister en une seule de ces opérations, ce qui revient à créer autant de niveaux que de contractions. Un ensemble de contractions topologiquement indépendantes peut également être opéré en une seule fois, engendrant alors un niveau grossier contenant environ 20% de sommets en moins.

Des simplifications successives peuvent être appliquées, "poussant" les triangles supprimés de plus en plus loin dans la hiérarchie. Il en résulte une structure topologique multirésolution ayant exactement les même propriétés que celles

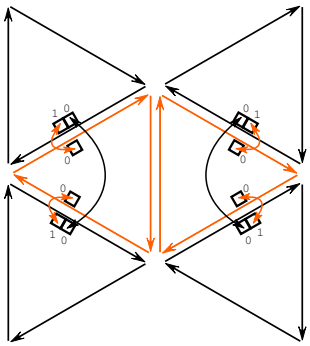


Figure 14: *Niveau 1 après la contraction d'arête*

construites par subdivision présentées auparavant. Les différents maillages correspondant aux niveaux intermédiaires peuvent être parcourus simplement et aussi efficacement qu'une structure de demi-arêtes classique. Les algorithmes faisant appel aux relations d'adjacence pour calculer de la géométrie peuvent donc être exécutés en temps optimal. Les "montées" et "descentes" de l'information géométrique peuvent être effectuées sans avoir à reconstruire la topologie des maillages intermédiaires, ceux-ci étant d'ores et déjà disponibles.

Les images 15 et 16 montrent des exemples d'application utilisant les demi-arêtes multirésolution dans ce cadre. On y voit en haut l'objet de départ; suivent un affichage filaire du maillage fin et de sa version la plus grossière. Sur la figure 15 les hautes fréquences ont été supprimées dans l'image de gauche et les fréquences moyennes exagérées dans la l'image de droite. Sur la figure 16 les fréquences moyennes sont exagérées dans l'image de gauche et des éditions multirésolution ont été appliquées dans l'image de droite.

6. Conclusion

Nous avons exposé dans cet article comment, par sa généralité et sa souplesse d'utilisation, une structure topologique telle que les cartes multirésolution peut être utilisée dans des contextes différents.

Dans le cadre des surfaces de subdivision multirésolution, en plus de son efficacité et de ses besoins en mémoire similaires aux structures classiques [KCB07a], ce modèle permet l'utilisation de schémas rarement utilisés dans ce contexte et qui peuvent alors y apporter leurs propres avantages. Dans le cadre des maillages progressifs, l'accès à toute l'information topologique permet une implantation efficace des algorithmes de filtrage ou d'édition multirésolution sur ce type de maillages. Nous menons actuellement une étude plus poussée de l'utilisation de notre modèle dans ce contexte.

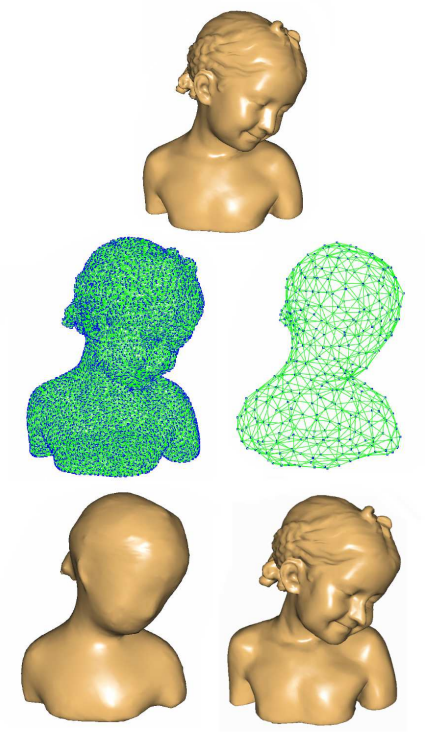


Figure 15: *Le modèle Bimba*

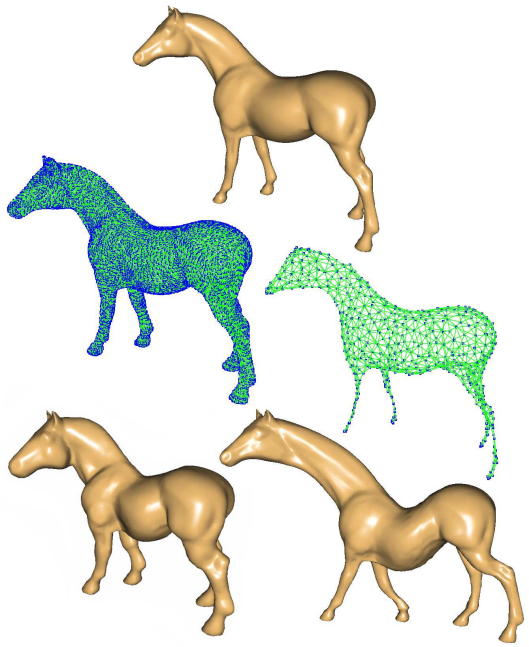


Figure 16: *Le modèle Horse*

Références

[Ber07] BERTRAM M.: Wavelet analysis for progressive meshes. In *Proceedings of SCCG'07* (2007).

[BK03] BRUN L., KROPATSCH W.: Combinatorial pyramids. In *IEEE International Conference on Image Processing (ICIP)* (Barcelona, Spain, 2003), vol. 2, pp. 33–36.

[CC78] CATMULL E., CLARK J.: Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355.

[DLG90] DYN N., LEVIN D., GREGORY J. A.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics 9*, 2 (1990), 160–169.

[GSDL06] GRASSET-SIMON C., DAMIAND G., LIENHARDT P.: nd generalized map pyramids: Definition, representations and basic operations. *Pattern Recognition 39*, 4 (2006), 527–538.

[GSS99] GUSKOV I., SWELDENS W., SCHRÖDER P.: Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH '99* (1999), pp. 325–334.

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of SIGGRAPH '96* (1996), pp. 99–108.

[KCB07a] KRAEMER P., CAZIER D., BECHMANN D.: Extension multirésolution des cartes combinatoires: application a la représentation de surfaces de subdivision multirésolution. *Revue Electronique Francophone d'Informatique Graphique*, 1 (2007).

[KCB07b] KRAEMER P., CAZIER D., BECHMANN D.: Multiresolution half-edges. In *Proceedings of SCCG'07* (2007).

[Kob00] KOBBELT L.: $\sqrt{3}$ subdivision. In *Proceedings of SIGGRAPH'00* (2000), pp. 103–112.

[Loo87] LOOP C.: *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah, 1987.

[LSS*98] LEE A., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: Maps: Multiresolution adaptive parameterization of surfaces. *Computer Graphics 32*, Annual Conference Series (1998), 95–104.

[PR00] PAJAROLA R., ROSSIGNAC J.: Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 79–93.

[PS04] PETERS J., SHIUE L.-J.: Combining 4- and 3-direction subdivision. *ACM Transactions on Graphics 23*, 4 (2004), 980–1003.

[SL03] STAM J., LOOP C.: Quad/triangle subdivision. *Computer Graphics Forum 22*, 1 (2003), 79–85.

[SW05] SCHAEFER S., WARREN J.: On c^2 triangle/quad subdivision. *ACM Transactions on Graphics 24*, 1 (2005), 28–36.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Triangulations Faiblement Contraintes

Mathieu Brévilliers, Nicolas Chevallier et Dominique Schmitt

Laboratoire MIA, Université de Haute-Alsace
4, rue des Frères Lumière, 68093 Mulhouse Cedex, France
{*Mathieu.Brevilliers, Nicolas.Chevallier, Dominique.Schmitt*} *@uha.fr*

Abstract

Given a set S of line segments in the plane, we introduce a new family of partitions of the convex hull of S called weakly constrained triangulations of S. The set of faces of such a triangulation is a maximal set of disjoint triangles whose open sides either do not cut S or are contained in S. The faces whose vertices are in three distinct segments of S allow to define a new kind of diagrams called segment triangulations of S. We study the relations between the weakly constrained triangulations and the segment triangulations and we show that several properties of planar point set triangulations extend to segment triangulations. We also show that, if S is in general position, there exists a unique segment triangulation of S whose faces are inscribable in circles whose interiors do not intersect S. This triangulation, called segment Delaunay triangulation, is dual to the segment Voronoi diagram. The main result of this paper is that the local optimality which characterizes point set Delaunay triangulations [Law77] extends to segment Delaunay triangulations.

Étant donné un ensemble S de segments de droite dans le plan, nous présentons une nouvelle famille de partitions de l'enveloppe convexe de S, que nous appelons triangulations faiblement contraintes de S. L'ensemble des faces d'une telle triangulation est un ensemble maximal de triangles disjoints dont les côtés ouverts soit sont contenus dans S, soit ne coupent pas S. Les faces dont les sommets sont répartis sur trois segments de S permettent de définir une nouvelle famille de diagrammes, appelés triangulations de segments de S. Nous étudions les relations entre les triangulations faiblement contraintes et les triangulations de segments et nous montrons que plusieurs propriétés des triangulations d'ensembles de points s'étendent aux triangulations de segments. Nous montrons aussi que si S est en position générale, alors il existe une unique triangulation de segments dont les faces sont inscriptibles dans des cercles dont les intérieurs ne coupent pas S. Cette triangulation, appelée triangulation de Delaunay de segments, est duale du diagramme de Voronoi de segments. Le résultat principal de cet article est que l'optimalité locale qui caractérise les triangulations de Delaunay d'ensembles de points [Law77] s'étend aux triangulations de Delaunay de segments.

Introduction

Étant donné un ensemble S de n points dans le plan, une triangulation de S est une décomposition de l'enveloppe convexe de S en triangles disjoints dont les sommets sont les points de S. Le problème de la construction d'une triangulation de S admet de nombreuses applications notamment pour la modélisation de surfaces, de terrains, ... Dans ces applications il s'agit de trouver la triangulation la plus régulière possible c'est-à-dire celle dont les triangles sont les plus "équilatéraux possibles". En 1978, Sibson [Sib78] a montré que la triangulation la plus régulière est la triangulation de Delaunay. Cette triangulation est telle que le cercle circonscrit à ses triangles ne contient aucun site de S en son intérieur [Del34]. Lorsqu'il s'agit de modéliser un terrain, celui-ci contient souvent des contraintes telles que des rivières, des routes, des crêtes de montagnes, ... et les triangles à construire ne doivent pas couper ces contraintes. Pour résoudre ce problème, Lee et Lin [LL86] ont introduit la notion de triangulation contrainte. Étant donnés un ensemble S de points et un ensemble T de segments qui relient certains de ces points, il s'agit d'une triangulation de S dont T est un sous-ensemble de côtés (voir Figure 1(a)). Lee et Lin ont montré que la triangulation contrainte la plus régulière est la triangulation contrainte de Delaunay qui est elle aussi carac-

térisée à l'aide des cercles circonscrits à ses faces. Cependant, si les segments sont particulièrement longs ou si les points sont mal disposés autour des segments, on remarque que les triangles s'appuyant sur les contraintes sont très irréguliers. Pour résoudre ce problème, une solution intéressante consiste à ajouter des points, appelés points de Steiner, dans l'ensemble initial puis à mettre à jour la triangulation contrainte en tenant compte de ces nouveaux points (voir, par exemple, [BE95]). En effet, les points de Steiner qui sont ajoutés sur des segments permettent de les décomposer en sous-segments et de construire de meilleures triangulations contraintes (voir Figure 1(b)). Avec cette méthode, la décomposition d'un segment contraint en sous-segments est donc la même des "deux côtés" de ce segment. Cela n'est pas toujours justifié car la répartition des points n'est pas la même de part et d'autre d'un segment et on pourrait souhaiter traiter ces deux côtés indépendamment. On pourrait, par exemple, trianguler indépendamment les deux pentes de part et d'autre d'une crête de montagne.

Dans cet article, nous initions l'étude d'une nouvelle triangulation d'un ensemble S de points et de segments disjoints du plan, que nous appellerons triangulation faiblement contrainte. Les éléments de S, points et segments, sont appelés des sites. Pour obtenir une triangulation faiblement contrainte de l'enveloppe convexe $conv(S)$ de S, il faut remplir $conv(S) \setminus S$ avec des triangles disjoints dont les sommets sont dans S et dont les côtés ouverts soit sont contenus dans S, soit ne coupent pas S. Dans une telle triangulation (voir Figure 1(c)), un point ajouté sur un segment ne scinde pas ce segment en deux mais devient un sommet de plusieurs triangles qui sont d'un même côté du segment. Comme les sites de S sont supposés disjoints, une triangulation faiblement contrainte de S admet deux types de triangles : ceux dont les sommets sont sur deux sites distincts et ceux dont les sommets sont sur trois sites distincts. Dans ce papier nous allons plus particulièrement étudier le deuxième type de triangles en montrant, notamment, que leur nombre est un invariant de S (c'est-à-dire qu'il ne dépend pas de la triangulation faiblement contrainte choisie).

Pour cela nous introduisons une nouvelle famille de diagrammes définis sur un ensemble S de points et de segments, que nous appellerons triangulations de segments de S. L'ensemble des faces d'une triangulation de segments de S est un ensemble maximal de triangles disjoints tel que les sommets de chaque triangle appartiennent à trois sites distincts de S et tel qu'aucun autre point de ces triangles ne soient dans S. Les arêtes d'une triangulation de segments sont les composantes connexes (éventuellement de dimension deux) de l'enveloppe convexe de S privée des faces ouvertes et des sites. Ces définitions sont naturelles car, si S est un ensemble de points, on retrouve les définitions des faces et des arêtes d'une triangulation d'un ensemble de points.

Dans les deux premières sections, nous étudions les relations entre les triangulations faiblement contraintes et les

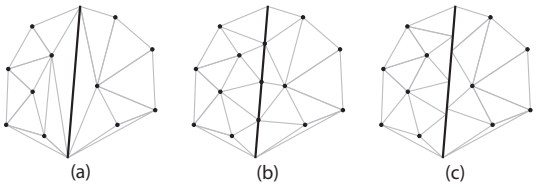


Figure 1: *Exemples de triangulation contrainte (a), de triangulation de Steiner (b) et de triangulation faiblement contrainte (c).*

triangulations de segments. Nous montrons également que plusieurs propriétés des triangulations d'ensembles de points s'étendent aux triangulations de segments.

Dans la section suivante, nous prouvons qu'il existe une unique triangulation de segments de S dont les faces sont inscriptibles dans des cercles vides. Nous montrons que cette triangulation, appelée triangulation de Delaunay de segments, est duale du diagramme de Voronoi de segments et qu'elle peut, par conséquent, être construite en temps $O(n \log n)$.

La triangulation de Delaunay d'un ensemble de points admet une importante caractérisation locale qui est utilisée pour prouver plusieurs de ses propriétés d'optimalité et qui permet de tester en temps linéaire si une triangulation donnée est de Delaunay. Selon cette caractérisation locale, une triangulation est de Delaunay si et seulement si tout couple de faces adjacentes à un même côté est en position de Delaunay relativement aux quatre sites qui les définissent [Law77]. Le résultat principal de la seconde partie de l'article est que cette propriété caractérise aussi la triangulation de Delaunay de segments parmi toutes les triangulations de segments d'un ensemble de segments donné. Nous donnons également une autre propriété locale qui caractérise l'ensemble des triangulations de segments qui ont la même topologie que la triangulation de Delaunay de segments.

1. Triangulations Faiblement Contraintes et Triangulations de Segments

Soit S un ensemble fini de $n \geq 2$ segments fermés disjoints dans le plan, appelés sites. Dans tout l'article, on considère qu'un segment fermé peut être réduit à un point. Nous disons qu'un cercle est tangent à un site s si s intersecte ce cercle mais pas son intérieur. Les sites de S sont en position générale, c'est-à-dire que nous supposons qu'il n'existe pas trois extrémités de segments qui sont alignées ni de cercle tangent à quatre sites.

Définition 1 *(i) Étant donné un ensemble S de sites, nous appelons S-polygone (éventuellement à trous) toute partie A de $conv(S)$ qui est fermée, de dimension deux, égale à la fermeture de son intérieur, telle que $A \setminus S$ est connexe et telle que la frontière de A est composée d'un nombre fini de seg-*

ments disjoints qui :
– soit sont fermés et inclus dans S,
– soit sont ouverts, ne coupent pas S et ont leurs extrémités dans S.
(ii) Nous appelons triangulation faiblement contrainte de A (relativement à S), toute partition de A en triangles dont les sommets sont dans S, dont les intérieurs ne coupent pas S et dont les côtés ouverts soit ne coupent pas S, soit sont inclus dans S (voir Figure 2).
Si $A = conv(S)$, une telle triangulation est appelée une triangulation faiblement contrainte de S.

Nous définissons maintenant la notion de triangulation de segments et nous montrons ensuite le lien avec la notion de triangulation faiblement contrainte.

Définition 2 *Une triangulation de segments P de S est une partition de l'enveloppe convexe $conv(S)$ de S en sites, faces et arêtes tels que :*
(i) Chaque face de P est un triangle ouvert dont les sommets appartiennent à trois sites distincts de S et dont les côtés ouverts ne coupent pas S,
(ii) Aucune face ne peut être ajoutée sans couper une face déjà existante,
(iii) Les arêtes de P sont les composantes connexes (éventuellement de dimension deux) de $conv(S) \setminus (F \cup S)$, où F est l'ensemble des faces de P.

Une telle triangulation existe toujours car, quel que soit l'ensemble *S*, il existe un nombre fini de faces qui vérifient la Définition 2. En effet, il est facile de remarquer qu'au maximum deux triangles disjoints peuvent avoir leurs sommets sur les trois mêmes sites (voir Figure 4(a)).

Le lemme suivant permet d'établir le lien entre les triangulations de segments et les triangulations faiblement contraintes.

Lemme 1 *Si A est un S-polygone qui coupe au moins trois sites de S alors toute triangulation faiblement contrainte de A contient au moins un triangle dont les sommets sont sur trois sites distincts de S.*

Preuve. Étant donnée une triangulation faiblement contrainte T de A, soit $\Delta_T(A)$ l'ensemble (éventuellement vide) des triangles de T qui ont un côté dans S. Nous montrons, par récurrence sur le nombre $|\Delta_T(A)|$ de triangles de $\Delta_T(A)$, que T contient au moins un triangle dont les sommets appartiennent à trois sites distincts de S.
Il est évident que, si $\Delta_T(A) = \emptyset$, les sommets de tous les triangles de T appartiennent à trois sites distincts. Supposons maintenant que le résultat est vrai pour toute triangulation faiblement contrainte T telle que $|\Delta_T(A)| < k$ ($k \geq 1$).
Comme A coupe au moins trois sites, pour toute triangulation faiblement contrainte T de A avec $|\Delta_T(A)| = k$ et pour tout triangle fermé t de $\Delta_T(A)$, la fermeture $\overline{A \setminus t}$ de $A \setminus t$ coupe les mêmes sites que A. Si $A \setminus t$ est connexe, $A' = \overline{A \setminus t}$ est un S-polygone. Sinon, $A \setminus t$ a deux composantes connexes et la fermeture de l'une des deux au moins est un S-polygone. Dans ce dernier cas, chacun des deux S-polygones

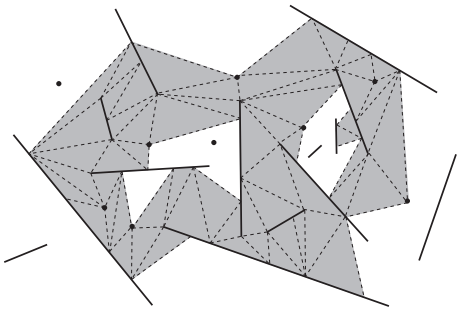


Figure 2: *Une triangulation faiblement contrainte (lignes en pointillés) d'un S-polygone (en gris).*

intersecte les deux sites auxquels appartiennent les sommets de t. Cela signifie qu'au moins l'un de ces S-polygones coupe au moins trois sites. Soit A' ce S-polygone. Dans les deux cas, si T' est la restriction de T à A', $|\Delta_{T'}(A')| < |\Delta_T(A)|$. Donc, d'après l'hypothèse de récurrence, T' contient au moins un triangle dont les sommets appartiennent à trois sites distincts de S. Il en est de même pour T. □

D'après ce lemme, pour toute triangulation faiblement contrainte de S, l'ensemble F de triangles dont les sommets sont sur trois sites distincts de S est maximal. En effet, la fermeture de toute composante connexe e de $conv(S) \setminus (F \cup S)$ est soit un segment qui relie deux points, soit un S-polygone. Dans ce dernier cas, d'après le Lemme 1, $\overline{e}$ ne peut intersecter que deux sites. C'est pourquoi aucun triangle dont les sommets sont sur trois sites distincts de S ne peut être ajouté sans couper $F \cup S$. On en déduit les théorèmes suivants :

Théorème 1 *Toute triangulation faiblement contrainte de S est un raffinement d'une triangulation de segments de S, c'est-à-dire une triangulation de segments dont les arêtes sont décomposées en triangles.*

Théorème 2 *La fermeture d'une arête d'une triangulation de segments de S coupe exactement deux sites de S.*

Cela montre qu'une arête d'une triangulation de segments P de S est vraiment une arête, dans ce sens qu'elle "connecte" exactement deux sites de S. Cela permet également de déduire directement la forme de l'arête. La fermeture d'une arête est soit un segment qui relie deux points de deux sites distincts de S, soit un triangle avec un côté et son sommet opposé dans S, soit un quadrilatère (pas forcément convexe) avec deux côtés opposés dans S (voir Figure 3). De plus, chaque arête de P contient :
– soit deux côtés de deux triangles de P,
– soit un côté d'un triangle de P et un côté de $conv(S)$ qui n'est pas un site,
– soit deux côtés de $conv(S)$ qui ne sont pas des sites.

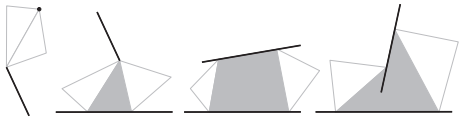


Figure 3: *Exemples d'arêtes (en gris) qui relient deux sites dans une triangulation de segments.*

2. Propriétés Topologiques des Triangulations de Segments

Comme chaque arête d'une triangulation de segments P de S "connecte" deux sites de S, nous pouvons associer à P un graphe abstrait tel que :
– les sommets du graphe sont les sites de S,
– les arêtes qui relient deux sites s et t dans le graphe sont les arêtes de P dont les fermetures coupent s et t.

Proposition 1 *Le graphe abstrait associé à une triangulation de segments P de S est planaire.*

Preuve. Pour tout site s de S, soit γ_s une courbe de Jordan convexe fermée telle que :
– s est à l'intérieur de γ_s (i.e. dans la partie du plan bornée par γ_s),
– $S \setminus s$ est à l'extérieur de γ_s,
– l'intérieur de γ_s intersecte uniquement les arêtes de P dont les fermetures coupent s.
On remplace maintenant chaque site s par un point p_s à l'intérieur de γ_s. Pour toute arête e de P qui intersecte γ_s, on remplace la partie de e à l'intérieur de γ_s par un segment qui relie p_s à un point de e sur γ_s. En faisant cela, l'ordre des arêtes autour de s reste inchangé et les arêtes réduites sont disjointes. Une fois cette transformation effectuée pour chaque courbe de Jordan γ_s, on remplace chaque arête réduite par un arc de Jordan inclus dans l'arête. Finalement, on obtient une représentation planaire G du graphe abstrait associé à P (voir Figure 4(b)). □

Théorème 3 *Toute triangulation de segments P d'un ensemble S de n sites contient $3n - n' - 3$ arêtes et $2n - n' - 2$ faces, où n' est le nombre de côtés de $conv(S)$ qui ne sont pas des sites.*

Preuve. Compter les arêtes et les faces de P revient à compter les arêtes et les faces bornées de la représentation planaire G construite dans la preuve de la Proprosition 1. De plus, la face non bornée de G correspond au complémentaire de $conv(S)$. Le résultat est donc une conséquence immédiate de la relation d'Euler, du fait que chaque face bornée de G a trois côtés et du fait que les arêtes adjacentes à une (respectivement aucune) face bornée apparaissent une fois (respectivement deux fois) en parcourrant la frontière de la face non bornée de G. □

Une conséquence intéressante de ce théorème est que la taille d'une triangulation de segments est linéaire avec le nombre de sites. De plus, cela montre que le nombre de triangles de la triangulation est un invariant de l'ensemble de sites. C'est une extension d'une propriété bien connue des triangulations d'ensembles de points dans le plan.
Il résulte du Théorème 1 que le nombre de triangles qui s'appuient sur trois sites dans une triangulation faiblement contrainte est également un invariant de l'ensemble de sites et ne dépend donc pas du nombre de points ajoutés sur les sites. Nous nous intéresserons ici au placement de ces triangles, à travers l'étude de triangulations de segments particulières.

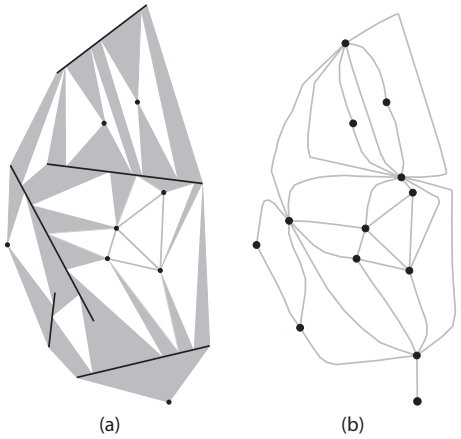


Figure 4: *Une triangulation de segments (a) (les sites sont en noir, les arêtes en gris, et les faces en blanc) et son graphe associé (b).*

En utilisant la représentation planaire G construite dans la preuve de la Proposition 1, on peut associer à la triangulation de segments P une carte combinatoire M :
– le graphe sous-jacent est le graphe abstrait associé à P,
– pour tout sommet s de M, l'ordre circulaire des arêtes issues de s correspond à l'ordre des arcs de Jordan autour de s (dans le sens trigonométrique) dans la représentation planaire G.
En général, la même carte M est associée à plusieurs triangulations de segments différentes. On dit que :

Définition 3 *Deux triangulations de segments de S ont la même topologie si elles ont la même carte combinatoire associée.*

Pour utiliser M comme structure de données pour stocker une triangulation de segments P, il suffit d'ajouter les coordonnées des sommets des triangles de P dans la structure : un sommet par arête orientée. L'espace nécessaire pour stocker une triangulation d'un ensemble S de n sites est donc en $O(n)$. Par ailleurs, comme une triangulation contrainte est un cas particulier de triangulation faiblement contrainte, il est possible de construire une triangulation faiblement contrainte en temps $O(n \log n)$ en utilisant un algorithme optimal

de construction d'une triangulation contrainte (par exemple, un algorithme par balayage [Ede00]). En outre, d'après le Théorème 1, toute triangulation contrainte de S est un raffinement d'une triangulation de segments de S. Donc ce même algorithme peut être facilement adapté pour construire une triangulation de segments en temps $O(n \log n)$.

3. Triangulation de Delaunay de Segments et Diagramme de Voronoi de Segments

Parmi l'ensemble de toutes les triangulations de segments, certaines sont remarquables : on pourrait s'intéresser, par exemple, aux triangulations de segments dont la somme des aires des faces est maximale. Dans cette section, nous nous intéressons plus particulièrement à la triangulation de segments dont les faces sont inscriptibles dans des cercles vides. Nous prouvons l'existence et l'unicité de cette triangulation de segments particulière et nous montrons qu'elle est duale du diagramme de Voronoi de segments (voir Figure 5). Nos preuves utilisent différentes propriétés du diagramme de Voronoi de segments qui peuvent être trouvées dans [AK98], [BY95] et [OBS92].

Soit F l'ensemble des triangles du plan tels que les sommets de chaque triangle appartiennent à trois sites distincts de S et tels que l'intérieur du cercle circonscrit à chaque triangle n'intersecte pas S.

Théorème 4 *(i) Les triangles de F sont les faces d'une triangulation de segments P de S appelée triangulation de Delaunay de segments.*
(ii) La carte combinatoire M associée à P est duale du diagramme de Voronoi de segments de S.

Preuve. Comme l'intérieur du cercle circonscrit à un triangle de F est vide, deux de ces triangles ne peuvent pas se couper. Donc ces triangles sont des faces d'une triangulation de segments. D'une part, d'après le Théorème 3, le nombre de triangles d'une triangulation de segments de S est égal au nombre de sommets du diagramme de Voronoi $Vor(S)$ de S. D'autre part, chaque sommet du diagramme de Voronoi correspond à un triangle de F. Donc le nombre de triangles de F est maximal, ce qui signifie que F est l'ensemble des triangles d'une triangulation de segments P. De plus, par définition du diagramme de Voronoi, les régions de $Vor(S)$ correspondent aux sites, qui sont, par définition, les sommets de M.
Il reste à étudier le cas des arêtes de M et de $Vor(S)$. Soit a une arête de $Vor(S)$ commune aux deux régions de Voronoi de s et t. Chaque point p de a est le centre d'un cercle vide C_p tangent aux sites s et t aux points p_s et p_t. Il est facile de prouver qu'un tel segment ouvert $p_s p_t$ ne rencontre jamais un triangle de F. Donc, pour chaque p de a, le segment ouvert $p_s p_t$ est inclus dans une arête de la triangulation de segments P. De plus, la réunion E_a de tous les segments ouverts $p_s p_t$, $p \in a$, est une partie connexe de $conv(S)$, donc E_a est inclus dans une seule arête e de P qui relie s à t. La dernière

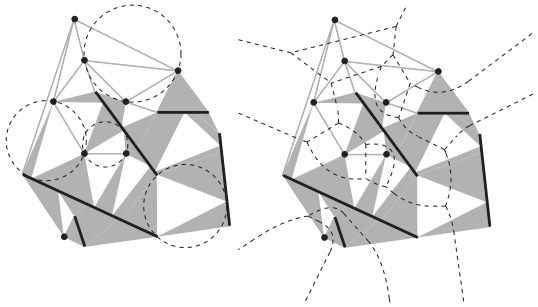


Figure 5: *Une triangulation de Delaunay de segments et une illustration de la dualité.*

chose à remarquer est que, pour chaque arête e de P, il existe exactement une arête a de $Vor(S)$ telle que $E_a \subset e$. Comme les nombres d'arêtes de P et de $Vor(S)$ sont égaux, il suffit de prouver que, pour chaque arête e de P, il existe au moins une arête a telle que $E_a \subset e$. Or, tout segment de la frontière d'une arête e qui relie s et t est du type $p_s p_t$ comme décrit précédemment. Donc il existe une arête a de $Vor(S)$ telle que $E_a \subset e$. □

Il est facile de voir que la triangulation de Delaunay de segments de S définie dans ce théorème est équivalente au dual de $Vor(S)$ introduit par Chew et Kedem [CK89]. Grâce à la dualité, en utilisant un algorithme qui construit le diagramme de Voronoi de segments, la triangulation de Delaunay de segments peut être calculée en temps $O(n \log n)$ [OBS92]. Une triangulation faiblement contrainte de S dont les faces qui s'appuient sur trois sites sont inscriptibles dans des cercles vides peut donc également être construite en temps $O(n \log n)$.

4. Légalité dans les Triangulations de Segments

La légalité des arêtes est une propriété intéressante de la triangulation de Delaunay d'un ensemble de points du plan. Considérons une arête d'une triangulation d'un ensemble de points, ainsi que ses deux triangles adjacents. Cette arête est illégale si un sommet de l'un de ces triangles est à l'intérieur du cercle circonscrit à l'autre triangle. Il est bien connu que la triangulation de Delaunay d'un ensemble de points est l'unique triangulation de cet ensemble de points qui n'a pas d'arête illégale. Dans la suite, nous allons prouver une propriété similaire dans le cadre des triangulations de segments.

Définition 4 *Soient e une arête d'une triangulation de segments, T l'ensemble des faces adjacentes à e ($|T| \leq 2$), et S' l'ensemble des sites qui contiennent les sommets des faces de T. On dit que e est légale si les cercles circonsrits aux triangles de T ne contiennent aucun points de S' en leurs intérieurs.*

Théorème 5 *La triangulation de Delaunay de segments de S est l'unique triangulation de segments de S dont toutes les arêtes sont légales.*

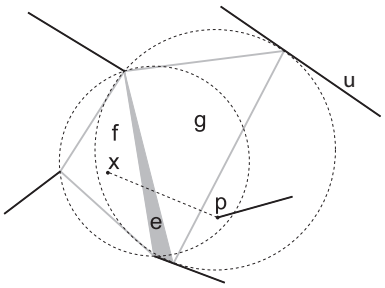


Figure 6: *Illustration de la preuve du Théorème 5.*

Preuve. Par définition, la triangulation de Delaunay de segments n'a pas d'arête illégale. Soit P une triangulation de segments qui n'est pas de Delaunay et soit f une face de P dont le cercle circonscrit c_f contient un point de S en son intérieur d_f. Il faut prouver que P a une arête illégale. Soient x un point de f et p un point de d_f qui est sur un site. On peut supposer que l'intérieur du segment xp ne coupe pas S. Soit k le nombre d'arêtes traversées par le segment xp. On peut remarquer que $k \geq 1$ car, par définition, p ne peut pas être dans f, ni dans une arête adjacente à f. Soient e la première arête traversée par xp, g l'autre face adjacente à e, c_g son cercle circonscrit, d_g l'intérieur de c_g, ab le côté de g contenu dans e et u le site qui contient le sommet de g qui n'est pas un sommet de e (voir Figure 6). Si $k = 1$, p est sur u et l'arête traversée par xp est donc illégale. Supposons maintenant que, si xp traverse k arêtes, alors au moins l'une d'entre elles est illégale. Il faut prouver que si xp traverse $(k+1)$ arêtes alors P a une arête illégale. Si l'arête e est illégale, c'est fini. Sinon, les points a et b ne peuvent pas être dans le disque d_f. De plus, le point $y = ab \cap xp$ est dans d_f. Donc le segment ab scinde d_f en deux parties. Soient d_1 la partie contenant la face f et d_2 l'autre partie. Le disque d_g contient forcément au moins d_1 ou d_2, et, comme e est légale, d_g ne peut pas contenir d_1. Il s'ensuit que le segment yp est dans d_g et traverse une arête de moins que xp. D'après l'hypothèse de récurrence, on peut conclure que P a une arête illégale. □

Comme nous l'avons remarqué dans la section 3, différentes triangulations de segments de S peuvent avoir la même topologie. En particulier, un nombre infini de triangulations de segments de S ont la même topologie que la triangulation de Delaunay de segments de S. Cependant, comme la triangulation de Delaunay de segments peut être facilement calculée quand sa topologie est connue, il n'est pas nécessaire de stocker les coordonnées de ses sommets qui sont, par ailleurs, souvent imprécises. Par conséquent, il est intéressant de savoir si une triangulation de segments de S donnée a la même topologie que la triangulation de Delaunay de S. De plus, supposons qu'une triangulation de segments de S soit de Delaunay : si on modifie légèrement la position des sites de S sans modifier la topologie de la triangulation, on peut se demander si la topologie est encore celle de la triangulation de Delaunay de segments pour ce nouvel ensemble S. Pour toutes ces raisons, nous définissons la légalité d'une arête dans le cadre des cartes associées aux triangulations de segments.

Définition 5 *Soit f une face d'une triangulation de segments de S. Le triangle de tangence de f est tel que :*
– ses sommets sont sur les trois mêmes sites que les sommets de f,
– l'intérieur de son cercle circonscrit ne coupe pas ces trois sites,
– si f et son triangle de tangence sont parcourrus dans le sens trigonométrique, ces trois sites sont rencontrés dans le même ordre.

Définition 6 *Soit M une carte associée à une triangulation de segments de S. Une arête e de M est légale dans les deux cas suivants :*

1. *e est adjacente à au plus un triangle interne.*
2. *e est adjacente à deux triangles internes T_1 et T_2 et la propriété suivante est vraie. Soient t, r, u, v les sites tels que t, r, u sont incidents à T_1 et r, t, v sont incidents à T_2 dans le sens trigonométrique. Soient $t_1r_1u_1$ et $r_2t_2v_2$ les triangles de tangence de T_1 et T_2 avec $t_i \in t$, $r_i \in r$, $u_1 \in u$ et $v_2 \in v$. Alors :*
   – le polygone $t_1t_2r_2r_1$ est soit réduit à un segment, soit un polygone simple orienté dans le sens trigonométrique (avec trois ou quatre côtés),
   – les intérieurs des cercles circonscrits à $t_1r_1u_1$ et $r_2t_2v_2$ ne coupent pas les sites t, r, u, v.

Théorème 6 *Soit M une carte associée à une triangulation de segments P de S. Si toutes les arêtes de cette carte sont légales, alors M est aussi la carte associée à la triangulation de Delaunay de segments de S.*

Preuve abrégée. Nous voulons prouver que la collection des triangles de tangence donne lieu à la triangulation de Delaunay de segments. En utilisant le théorème précédent, nous remarquons que la seule chose à prouver est que les intérieurs des triangles de tangence sont les faces d'une triangulation de segments de S.
La principale idée de la preuve est d'utiliser un résultat de Devillers et al. [DLPT98] selon lequel une représentation d'une carte combinatoire par des courbes lisses dans le plan est un graphe planaire si :
– tous les circuits de la carte sont représentés par des courbes fermées simples,
– l'ordre en chaque sommet s de la carte est donné par l'ordre géométrique des courbes issues du point qui représente s.
En fait, le résultat de Devillers et al. est formulé avec des segments au lieu de courbes lisses, mais avec un argument d'approximation on obtient le même résultat avec des courbes lisses.
Pour $\varepsilon > 0$ suffisamment petit, il est possible de construire un graphe planaire comme celui réalisé dans la Figure 7(a).

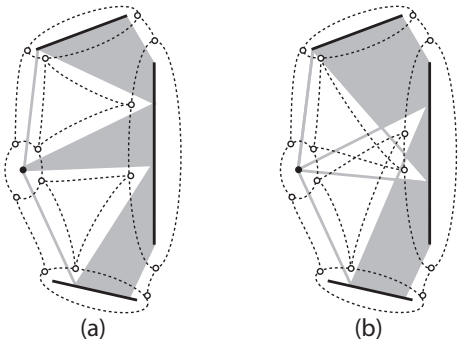


Figure 7: *(a) Graphe planaire déduit de P. (b) Une nouvelle représentation de la carte M'.*

Toutes les arêtes de ce graphe sont des courbes lisses qui sont soit à une distance inférieure à ε des sites, soit à une distance inférieure à ε des côtés des triangles de P. Ce graphe planaire est une représentation dans le plan d'une nouvelle carte combinatoire M' qui ne dépend pas de ε.

Maintenant, en déplaçant tous les triangles T de P jusqu'à leurs positions de tangence T', nous pouvons définir une nouvelle représentation de la carte M' :
– les courbes associées à chaque triangle de P se déplacent des triangles initiaux jusqu'aux triangles de tangence.
– les nouvelles courbes fermées autour des sites sont légèrement plus compliquées à définir. Supposons que T_1 et T_2 sont deux triangles adjacents de P qui ont chacun un sommet sur un site s. Soit γ_s "l'ancienne" courbe autour de s. Il existe un point p_i sur γ_s associé au sommet de T_i qui est sur s et il existe un point p_i' sur γ_s associé au sommet du triangle de tangence T_i' qui est sur s. Dans la nouvelle représentation de la carte M', nous considérons la partie de la courbe γ_s qui va de p_1' à p_2' en tournant autour de s dans le même sens que la partie de γ_s qui va de p_1 à p_2 (voir Figure 7(b)).
Cette définition assure que l'ordre géométrique des courbes issues d'un sommet est le même dans l'ancienne et dans la nouvelle représentation de la carte M'. Finalement, grâce à la légalité des arêtes, on peut prouver que la nouvelle représentation des circuits de M' sont des courbes fermées simples. Donc, d'après le résultat de Devillers et al., la nouvelle représentation de M' est un graphe planaire. Si on fait tendre ε vers zéro, on voit que les triangles de tangence sont les faces d'une triangulation de segments. □

Le Théorème 6 permet de tester si une triangulation de segments a la topologie de la triangulation de Delaunay de segments en vérifiant la légalité des arêtes. D'après le Théorème 3, le nombre d'arête est en $O(n)$, où $n = card(S)$, et le test peut donc être effectué en temps $O(n)$. D'où :

Corollaire 1 *Il existe un algorithme linéaire qui vérifie si une triangulation de segments donnée a la topologie de la triangulation de Delaunay de segments.*

Par dualité, cela permet de vérifier en temps linéaire l'exactitude de la topologie d'un diagramme de Voronoi de segments calculé par un programme. Pour plus de détails sur les vérifications efficaces de programmes en géométrie algorithmique, consulter, par exemple, [DLPT98] et [MNS*96].

Conclusion

Dans cet article, nous avons notamment prouvé que la triangulation de Delaunay de segments est l'unique triangulation de segments dont toutes les arêtes sont localement de Delaunay. Dans le cadre des triangulations d'ensembles de points, la notion de légalité des arêtes est à l'origine de la propriété d'équiangularité de la triangulation de Delaunay ainsi que d'un algorithme dit "de flip" [Law77] qui permet de transformer une triangulation quelconque en triangulation de Delaunay par une suite d'améliorations locales. Cet algorithme est particulièrement simple à implémenter et, bien qu'ayant une complexité en $O(n^2)$ dans le pire des cas, il est efficace en pratique lorsque la triangulation initiale n'est pas "trop mauvaise".
Les résultats de cet article devraient permettre de prouver des propriétés d'optimalité de la triangulation de Delaunay de segments et de donner un algorithme de flip pour construire la triangulation de Delaunay de segments à partir d'une triangulation de segments quelconque. En plus de la caractérisation locale que nous avons démontrée, il y a un indice qui nous fait penser qu'une sorte d'algorithme de flip devrait fonctionner avec les triangulations de segments. En considérant le relèvement de l'ensemble de sites S sur le paraboloide $z = x^2 + y^2$, il est facile de voir que les triangles de la triangulation de Delaunay de segments sont exactement les projetés des faces triangulaires de l'enveloppe convexe inférieure du relèvement de S. De plus, le relèvement de toute face qui n'est pas de Delaunay est au-dessus de cette enveloppe convexe inférieure, comme dans le cadre des triangulations d'ensembles de points.
Par ailleurs, une fois établies des propriétés d'optimalité de certaines triangulations de segments, on pourra s'intéresser à la régularité des triangulations faiblement contraintes. Il faudra alors établir diverses stratégies de raffinement d'une triangulation de segments pour obtenir une triangulation faiblement contrainte la plus régulière possible. Puis, les résultats pourront ensuite être comparés à ceux obtenus par des méthodes éprouvées de génération de maillages (voir, par exemple, [Rup95], [EG01] et [HPU05]).

D'autre part, très récemment, une attention toute particulière a été portée sur l'étude du diagramme de Voronoi d'un ensemble de segments en trois dimensions [MTT05], [SW06], [KS03], ... Cependant, la topologie de ce diagramme est connue uniquement pour un ensemble de trois droites [ELLD07]. L'étude du diagramme de Voronoi d'un ensemble de points a été grandement facilité par l'étude et la compréhension de son dual, le diagramme de Delaunay. Rappelons que, le diagramme de Delaunay de S a

plusieurs propriétés d'optimalité dont certaines restent valables en dimensions supérieures [Raj94], [SS99]. Jusqu'à maintenant, aucune propriété équivalente n'a été donnée, même dans le plan, pour le dual du diagramme de Voronoi de segments qui a été introduit par Chew et Kedem [CK89]. Plus surprenant encore, aucune famille de diagrammes contenant ce dual n'avait été définie alors même que plusieurs généralisations des triangulations d'ensembles de points ont déjà été étudiée : triangulations contraintes [LL86], pseudo-triangulations [RSS], pre-triangulations [AAH06], ... La notion de triangulation de segments comble ce vide et nous espérons que ces triangulations pourront être définies en dimension supérieure et que leur étude permettra une meilleure compréhension de la structure topologique du diagramme de Voronoi de segments en dimension supérieure.

References

[AAH06] AICCHHOLZER O., AURENHAMMER F., HACKL T.: Pre-triangulations and liftable complexes. In *Proc. 22th Annu. ACM Sympos. Comput. Geom.* (2006), pp. 282–291.

[AK98] AURENHAMMER F., KLEIN R.: Voronoi diagrams. In *Handbook of Computational Geometry*, Sack J.-R., Urrutia J., (Eds.). Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1998.

[BE95] BERN M. W., EPPSTEIN D.: Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, Du D.-Z., Hwang F. K.-M., (Eds.), second ed., no. 4 in Lecture Notes Series on Computing. World Scientific, 1995, pp. 47–123.

[BY95] BOISSONNAT J.-D., YVINEC M.: *Géométrie algorithmique*. Ediscience international, Paris, 1995.

[CK89] CHEW L. P., KEDEM K.: Placing the largest similar copy of a convex polygon among polygonal obstacles. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.* (1989), pp. 167–174.

[Del34] DELAUNAY B.: Sur la sphère vide. *Bull. Acad. Sci. USSR: Class. Sci. Math. Nat. 7* (1934), 793–800.

[DLPT98] DEVILLERS O., LIOTTA G., PREPARATA F. P., TAMASSIA R.: Checking the convexity of polytopes and the planarity of subdivisions. *Comput. Geom. Theory Appl. 11* (1998), 187–208.

[Ede00] EDELSBRUNNER H.: Triangulations and meshes in computational geometry. *Acta Numerica* (2000), 133–213.

[EG01] EDELSBRUNNER H., GUOY D.: Sink-insertion for mesh improvement. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry* (New York, NY, USA, 2001), ACM Press, pp. 115–123.

[ELLD07] EVERETT H., LAZARD S., LAZARD D., DIN M. S. E.: The voronoi diagram of three lines. In *SCG '07: Proceedings of the twenty-third annual symposium on Computational geometry* (New York, NY, USA, 2007), ACM Press, pp. 255–264.

[HPU05] HAR-PELED S., ÜNGÖR A.: A time-optimal delaunay refinement algorithm in two dimensions. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry* (New York, NY, USA, 2005), ACM Press, pp. 228–236.

[KS03] KOLTUM V., SHARIR M.: Three dimensional euclidean voronoi diagrams of lines with a fixed number of orientations. *SIAM J. Comput. 32*, 3 (2003), 616–642.

[Law77] LAWSON C. L.: Software for C^1 surface interpolation. In *Math. Software III*, Rice J. R., (Ed.). Academic Press, New York, NY, 1977, pp. 161–194.

[LL86] LEE D. T., LIN A. K.: Generalized Delaunay triangulation for planar graphs. *Discrete Comput. Geom. 1* (1986), 201–217.

[MNS*96] MEHLHORN K., NÄHER S., SCHILZ T., SCHIRRA S., SEEL M., SEIDEL R., UHRIG C.: Checking geometric programs or verification of geometric structures. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.* (1996), pp. 159–165.

[MTT05] MOURRAIN B., TÉCOURT J.-P., TEILLAUD M.: On the computation of an arrangement of quadrics in 3d. *Comput. Geom. Theory Appl. 30*, 2 (2005), 145–164.

[OBS92] OKABE A., BOOTS B., SUGIHARA K.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[Raj94] RAJAN V. T.: Optimality of the Delaunay triangulation in R^d. *Discrete Comput. Geom. 12* (1994), 189–202.

[RSS] ROTE G., SANTOS F., STREINU I.: Pseudo-triangulations - a survey. *Discrete Comput. Geom.* to appear.

[Rup95] RUPPERT J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms 18*, 3 (1995), 548–585.

[Sib78] SIBSON R.: Locally equiangular triangulations. *Comput. J. 21*, 3 (1978), 243–245.

[SS99] SCHMITT D., SPEHNER J.-C.: Angular properties of Delaunay diagrams in any dimension. *Discrete Comput. Geom. 5* (1999), 17–36.

[SW06] SCHÖMER E., WOLPERT N.: An exact and efficient approach for computing a cell in an arrangement of quadrics. *Comput. Geom. Theory Appl. 33*, 1–2 (2006), 65–97.

Compte rendu d'une mission à Siggraph 2007

Nicolas Courty[1] et Basile Sauvage[2]

[1]Université de Bretagne Sud, VALORIA/Samsara, Vannes
[2]Université Louis Pasteur, LSIIT, Strasbourg
E-mails : courty@univ-ubs.fr,Sauvage@dpt-info.u-strasbg.fr

Avant-propos

Nous sommes deux à avoir été choisis par un comité de l'AFIG pour participer à Siggraph qui avait lieu cette année à San Diego, Californie. Siggraph représente sans doute un des temps les plus forts de l'année pour la communauté d'informatique graphique. Plus qu'une simple conférence scientifique, Siggraph attire en nombre tous les acteurs de cette communauté (chercheurs, industriels, artistes, investisseurs, etc.), et leur permet d'interagir. Sur le plan de la recherche, Siggraph se distingue par l'excellence de son contenu (articles, notes de cours) et la qualité de ses présentations orales.

Nous avons choisi la structure suivante pour ce compte-rendu : dans un premier temps nous allons décrire l'organisation de la conférence. Dans un deuxième temps, une synthèse scientifique plus poussée sera faite sur nos deux thématiques de prédilection (animation et traitement de la géométrie). Finalement nous ébaucherons un panel de technologies commercialisées ou en développement présentées cette année.

1. Déroulement de la conférence

Siggraph est d'abord marquée par le gigantisme : plus de 24 000 participants cette année. Elle est ensuite marquée par la diversité : diversité des acteurs qui s'y retrouvent, et diversité des thématiques représentées, à l'image des nombreuses communautés qui composent l'informatique graphique.

Chacun a ses espaces d'exposition et de parole. Les chercheurs retrouvent leurs motifs habituels : les cours, plutôt concentrés en début de semaine; les articles, longs et courts; les posters. Les industriels préfèrent les stands d'exposition, les salles de réunion, mais ils ont aussi la possibilité de faire des présentations. Ces deux acteurs se mêlent dans un hall très prisé *emerging technologies* où sont présentées des technologies novatrices, pas nécessairement commercialisées. Le principal évènement artistique est le festival de films d'animations qui diffuse en continu pendant les cinq jours. Souvent en fin de journée, des groupes de travail ouverts et des tables rondes variées sont organisées sur des thèmes précis afin de favoriser les échanges.

Notons que les organisateurs ont annoncé la création d'une conférence Siggraph Asia qui aura lieu en décembre 2008 à Singapour †. Les proceedings de cette édition asiatique seront eux aussi publiés dans un numéro spécial de la revue ACM *Transaction on Graphics*. L'édition 2008 du Siggraph aura lieu quant à elle à Los Angeles à partir du 11 août 2008 ‡.

2. Synthèse scientifique

2.1. Animation de personnages

La première session d'animation de personnages de cette édition du Siggraph a mis l'accent sur la notion de contrôle de personnages. Cette problématique est sans doute motivée par les éditeurs de jeux vidéo qui, de plus en plus dans leurs productions, utilisent de grosses quantités de mouvements capturés et portent une attention particulière à la jouabilité de leurs jeux. Celle-ci impose d'avoir une adéquation importante entre le mouvement voulu par le joueur et le mouvement produit par le système d'animation. Cooper et al proposent dans [CHP07] un schéma d'apprentissage de ce type de contrôleur par apprentissage actif, c'est à dire par une sélection manuelle durant la période de capture des mouvements pertinents du point de vue du contrôleur. Dans [TLP07], Treuille et ses collègues proposent une méthode d'animation basée données qui construit un ensemble de contrôleurs à partir d'une tâche donnée. Les résultats présentés utilisent des mouvements de marche et

† http://www.siggraph.org/asia2008/index.htm
‡ http://www.siggraph.org/s2008/

d'évitement d'obstacles. La qualité des résultats est vraiment impressionnante, et semble parfaitement adaptée aux problématiques de jeux vidéos. Dans un registre équivalent, le travail de McCann et Pollard [MP07] propose de synthétiser un mouvement continu à partir de plusieurs petits mouvements capturés en fonction des entrées d'un utilisateur. Leur système repose sur un modèle des mouvements de l'utilisateur établi par apprentissage par renforcement. Les résultats montrés étaient beaucoup moins convaincants que le précédent article, et ce même si l'idée de disposer d'un modèle des interactions de l'utilisateur est intéressante. Finalement dans [CH07], Chai et Hodgins proposent une méthode pour construire un mouvement complet à partir de contraintes (postures clés, contraintes cinématiques, etc.) spécifiées par l'utilisateur. Leur méthode repose sur l'utilisation d'un modèle statistique dynamique élaboré au travers d'une phase d'apprentissage sur une base de données de mouvements. Si la formulation de ce problème récurrent en animation est originale, les résultats sont un peu décevants, et on peut se questionner sur l'efficacité de la méthode lorsque les postures clés ne sont pas choisies dans la base qui sert à l'apprentissage (capacité d'extrapolation). À titre anecdotique, on notera que les quatre articles de cette session ont été réalisés par seulement deux groupes de recherche, à savoir le groupe de Jessica Hodgins (Carnegie Mellon University) ainsi que celui de Zoran Popović (Washington University).

La deuxième session d'animation de personnage s'est montrée plus hétérogène dans les thèmes présentés. Dans [SKL07, YLvdP07] le problème de la simulation physique de la marche est traité. Ce problème relativement vieux et extrêmement connu dans la communauté roboticienne est ici abordé du point de vue du contrôle. Yin et al. [YLvdP07] proposent une stratégie d'élaboration de contrôleur relativement simple et intuitive qui s'avère donner des résultats surprenamment bons (une applet de démonstration est disponible sur leur site web). Sok et ses collègues de la Séoul National University [SKL07] proposent une méthode d'optimisation permettant de transformer un mouvement capturé ou un mouvement spécifié à la main par un animateur en un mouvement physiquement réalisable par un système mécanique ainsi qu'une méthode d'apprentissage de contrôleur. Leur première contribution est à mon sens très originale et très intéressante, et ce même si on peut avoir du mal à évaluer l'applicabilité de leur méthode pour un personnage $3D$ (les résultats présentés étaient exclusivement en $2D$). Dans [SH07], Safonova *et al.* approfondissent la méthode des *motion graphs* en proposant des algorithmes de construction prenant en compte la qualité des chemins disponibles ainsi qu'une méthode d'interpolation entre chemins candidats du graphe pour optimiser le résultat final. Bien que peu original, cet article propose des résultats tout à fait convaincants. Finalement Guenter [Gue07] de Microsoft Research présente un langage (D^*) de calcul formel au travers d'applications dans le domaine de l'animation (dynamique inverse, optimisation spatio-temporelle). Dédiée à la mécanique sous-jacente de son langage, cette présentation ne semblait pas vraiment destinée au publique de cette session.

2.2. Déformation

Le terme "déformation" est considéré ici dans une acception large. De nombreuses opérations géométriques sur les surfaces en général, et les maillages en particulier, peuvent être apparentées à des déformations : édition, animation, morphing, lissage, etc. Nous ne prétendons ici ni classifier exhaustivement ni décrire en détail les articles sur ce thème. Nous proposons plutôt une lecture partielle, d'un point de vue : celui des espaces de travail choisis.

Afin de mieux caractériser ces déformations, de nombreuses méthodes effectuent les calculs dans un espace de travail autre que l'espace euclidien de plongement. En d'autres mots, la géométrie est exprimée autrement que par des positions dans l'espace $\mathbb{R}^3$. La procédure classique consiste à i) envoyer l'objet géométrique dans l'espace de travail, ii) résoudre le problème (i.e. calculer la déformation) dans cet espace, et iii) revenir dans l'espace euclidien. L'idée générale est de transcrire le problème dans un espace où il sera plus facile à résoudre, parce que la formulation est plus simple, ou bien parce que la complexité de la résolution est moindre, ou encore parce que cet espace possède une métrique plus adaptée, etc. Cela cause souvent deux difficultés. D'une part le retour dans l'espace euclidien n'est pas forcément trivial, ni exact. D'autre part l'interaction peut être compliquée par le fait que l'expression de contraintes intuitives dans $\mathbb{R}^3$ (e.g. la position d'un sommet) est complexe dans ces espaces.

Un espace souvent utilisé est celui des coordonnées différentielles. Leur principe est de représenter le maillage par ses dérivées premières (repère local) ou secondes (son Laplacien), et de contraindre ces dérivées. Une propriété souvent mise en avant est leur propension à conserver les détails. De plus, l'écriture et la minimisation de certaines énergies utilisant les dérivées (e.g. tension, torsion) devient plus simple. Zhou *et al.* [ZHX*07] proposent de calculer sur GPU des déformations de surfaces de subdivision, en utilisant les coordonnées différentielles de leur maillage de contrôle. Au *et al.* [AFTCO07] manipulent les isocontours d'un champ scalaire sur le maillage pour contrôler efficacement les coordonnées différentielles. Shi *et al.* [SZT*07] les combinent avec diverses contraintes géométriques dans un modèle de skinning. Xu *et al.* [XZY*07] en proposent une généralisation pour les maillages animés.

Mitra *et al.* [MGP07] utilisent un espace appelé "transformation space" pour "symétriser" des objets 3D. Fondamentalement il s'agit d'un paramétrage des similitudes de $\mathbb{R}^3$. L'intérêt principal est de représenter les symétries simplement, et de disposer d'une notion de distance pertinente. Les

objets sont rendus plus symétriques par des contractions et des fusions de clusters dans cet espace.

Kilian *et al.* [KMP07] proposent essentiellement de faire des calculs de géodésiques dans un espace de grande dimension appelé "shape space", qui est l'ensemble des plongements possibles pour un maillage de connectivité donnée. Il peuvent ainsi créer et contrôler diverses opérations géométriques apparentées à des déformations. Un intérêt de leur méthode est de construire dans cet espace une métrique du "degré d'isométrie" entre deux positions, ce qui leur permet de calculer des déformations "aussi isométriques que possible".

Sumner *et al.* [SSP07] proposent de déformer l'espace ambiant à l'aide d'un "deformation graph" dont le plongement dans $\mathbb{R}^3$ approxime l'objet initial. L'espace de travail regroupe les transformations affines pour tous les nœuds du graphe. Cet espace est suffisamment grand pour permettre de bien mesurer la distorsion géométrique induite par la déformation. Il est suffisamment petit pour qu'une optimisation d'énergie non-linéaire soit réalisable en temps réel.

3. Synthèse technologique

240 exposants étaient présents dans la partie "trade floor" cette année, à savoir l'exposition centrale du Siggraph (voir l'image 4.e). De nombreux constructeurs étaient présents (Hewlett-Packard, NVidia) pour présenter leurs dernières versions de stations de travail ou de cartes graphiques, mais aussi de nombreux éditeurs de logiciels tels que Autodesk ou Adobe. Les studios de production étaient également largement représentés (tels que Sony Pictures Imageworks, Dreamworks, ou bien encore Blue Sky). En plus de la présentation de leurs derniers films (illustrés par des *panels* techniques faisant intervenir les lead designers, avec à l'honneur cette année *Ratatouille* et *Spiderman 3*), ce sont de véritables séances de recrutement qui animent chacun de ces stands.

Il est à noter que de nombreux périphériques de capture sont présentés sur les stands ; aussi bien de la capture de forme que de la capture de mouvements. Ainsi de nombreux scanner 3D sont exposés, ayant la plupart du temps une forme de pistolet émettant un faisceau balayant le modèle. Les précisions obtenus ainsi que les méthodes de reconstruction de maillages (ou de rendu basé points) permettent d'obtenir des résultats parfaitement utilisables dans un contexte de production. Associées à ces technologies, on notera aussi la présence sur les stands d'imprimantes tridimensionnelles qui visiblement ont aussi atteint un degré de maturité suffisant pour leur exploitation commerciale. Dans le cadre de la capture du mouvement, on assiste là aussi à une explosion de l'offre, sans doute en réponse au prix très important de l'équipement proposé par le leader du marché VICON. En plus des traditionnelles méthodes utilisant des marqueurs passifs ou actifs (diodes), de nouvelles technologies sont déployées : combinaisons truffées d'accéléromètres et liaisons wifi avec une base (ce qui autorise des lieux de capture moins contraints) ou bien encore par utilisation de capture vidéo (avec la société française RealViz). On ressent aussi cette préoccupation avec une session d'articles *Performance Capture* où de nouvelles méthodes de capture moins contraignantes ont été présentées [BBA*07, WCF07, VAV*07, RND*07].

Ils nous a aussi semblé qu'il y avait cette année une véritable volonté d'ouvrir la communauté Siggraph vers les problèmes d'interaction humain-machine. De nombreuses tables ou périphériques multitactiles étaient présents (dont notamment celle du centre de recherche de Mitsubishi MERL ou bien la très médiatisée *Surface* de Microsoft). On notera aussi la présence de murs ou de globes d'interactions (Globe4D, déjà présenté à Laval Virtual) dans la partie *Emerging technologies* ainsi qu'un nouveau cours dédié sur le sujet (*Interaction Tomorrow*). Les aspects visualisation étaient eux aussi énormément représentés, avec notamment un écran 3D haute définition assez impressionnant utilisant la stéréographie, ou bien encore le périphérique de visualisation omnidirectionnel basé sur un miroir rotatif et dont la partie théorique a été exposée dans l'article [JMY*07].

4. Conclusion

À l'issue de cette manifestation on ne peut qu'encourager nos collègues à chercher à y participer. Le diversité des thèmes abordés, des formes de communication, et le foisonnement des échanges sont extrêmement stimulants sur le plan scientifique. La qualité générale des travaux exposés et la possibilité d'échanger avec leurs auteurs en fait une occasion d'observer les tendances scientifiques actuelles, et, en même temps, d'y trouver des idées précises.

Remerciements

Nous tenons à remercier chaleureusement le comité de direction de l'AFIG ainsi que le GDR IG pour nous avoir offert l'opportunité d'assister à Siggraph.

References

[AFTCO07] AU O. K.-C., FU H., TAI C.-L., COHEN-OR D.: Handle-aware isolines for scalable shape editing. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Shape Deformation.

[BBA*07] BICKE B., BOTSCH M., ANGST R., MATUSIK W., OTADUY M., PFISTER H., GROSS M.: Multi-scale capture of facial geometry and motion. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Performance Capture.

[CH07] CHAI J.-X., HODGINS J.: Constraint-based motion optimization using a statistical dynamic model. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation I.

[CHP07] COOPER S., HERTZMANN A., POPOVIĆ Z.: Active learning for real-time motion controllers. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation I.

[Gue07] GUENTER B.: Efficient symbolic differentiation for graphics applications. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation II.

[JMY*07] JONES A., MCDOWALL I., YAMADA H., BOLAS M., DEBEVEC P.: Rendering for an interactive 360-degree light field display. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Light Field & High-Dynamic-Range Imaging.

[KMP07] KILIAN M., MITRA N. J., POTTMANN H.: Geometric modeling in shape space. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Geometry Processing II.

[MGP07] MITRA N. J., GUIBAS L. J., PAULY M.: Symmetrization. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Geometry Processing II.

[MP07] MCCANN J., POLLARD N.: Responsive characters from motion fragments. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation I.

[RND*07] RASKAR R., NII H., DEDECKER B., HASHIMOTO Y., SUMMET J., MOORE D., ZHAO Y., WESTHUES J., DIETZ P., INAMI M., NAYAR S., BARNWELL J., NOLAND M., BEKAERT P., BRANZOI V., BURNS E.: Prakash: Lighting-aware motion capture using photosensing markers and multiplexed illumination. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Performance Capture.

[SH07] SAFONOVA A., HODGINS J.: Construction and optimal search of interpolated motion graphs. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation II.

[SKL07] SOK K., KIM M., LEE J.: Simulating biped behaviors from human motion data. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation II.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Shape Deformation.

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Shape Deformation.

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation I.

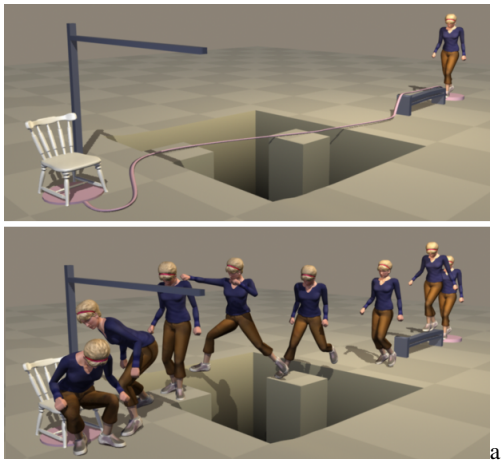
[VAV*07] VLASIC D., ADELSBERGER R., VANNUCCI G., BARNWELL J., GROSS M., MATUSIK W., POPOVIĆ J.: Practical motion capture in everyday surroundings. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Performance Capture.

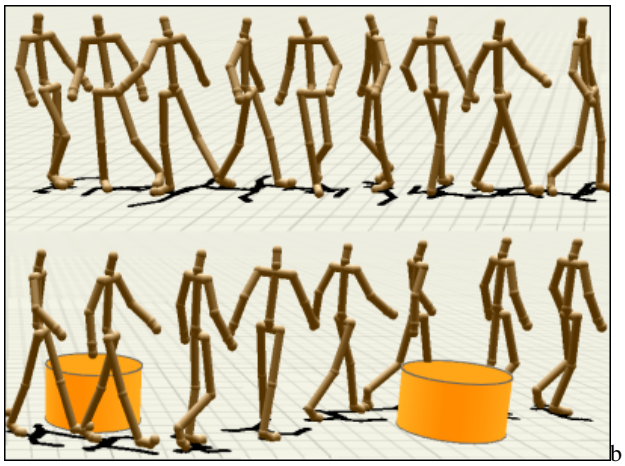
[WCF07] WHITE R., CRANE K., FORSYTH D.: Capturing and animating occluded cloth. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Performance Capture.

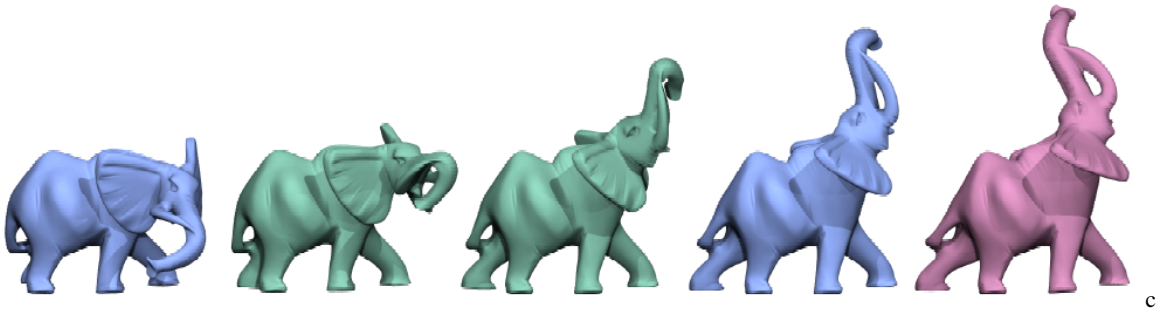
[XZY*07] XU W., ZHOU K., YU Y., TAN Q., PENG Q., GUO B.: Gradient domain editing of deforming mesh sequences. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Shape Deformation.

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Character Animation II.

[ZHX*07] ZHOU K., HUANG X., XU W., GUO B., SHUM H.-Y.: Direct manipulation of subdivision surfaces on gpus. *ACM Tra. on Graphics (proceedings of Siggraph 2007) 26*, 3 (Aug. 2007). session : Graphics Architecture.


a


b


c

d


e

Figure 1: *(a) Interpolation de chemins dans des graphes de mouvement par Safanova et al. [SH07] (b) Contrôleur de mouvement de marche par Treuille et al. [TLP07] (c) Interpolation et extrapolation dans [KMP07] (d) Poster de l'édition 2007 du Siggraph (e) Photo d'ensemble du* trade floor.

Stylisation d'objets éclairés par des cartes d'environnement HDR

Romain Vergne et Xavier Granier

INRIA LaBRI Universités de Bordeaux

Abstract

In this paper, we introduce a pipeline for an interactive and stylized rendering of High-Dynamic Range (HDR) environment-mapped objects. By using stylization on HDR images, the quality, the segmentation and the details are improved. Furthermore, this pipeline allows an easy combination of 2D stylization (on the environment map, and on the images) and 3D stylization (on the object). Based on this pipeline, we present new interactive styles that illustrate our approach. Through these styles, we detail the purpose and the action of each rendering step, in order to show its flexibility and to allow the reader the implementation of new styles.

Dans cet article, nous introduisons un pipeline de rendu permettant de styliser de manière interactive des objets éclairés par des cartes d'environnement à grande dynamique (High-Dynamic Range ou HDR). L'utilisation d'images HDR permet d'améliorer la qualité de certains traitements, comme les segmentations ou l'extraction des détails. De plus, cette architecture permet de combiner facilement des stylisations 2D (sur la carte d'environnement et sur les images) et 3D (sur les objets). Les nouveaux styles que nous présentons sont basés sur ce pipeline et illustrent la flexibilité de notre approche.

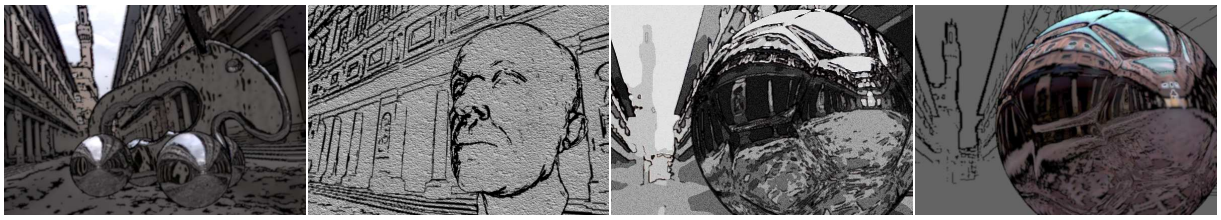
1. Introduction

Une carte d'environnement représente une solution simple et efficace pour simuler un éclairement distant. Un interêt croissant s'est porté sur cette technique dans la communauté scientifique durant les dernières années (recherche sur le précalcul du transfert de la radiance [SKS02]) dans le but de développer des effets d'éclairement complexes et réalistes. Les cartes d'environnement sont aussi un outil de modélisation simple pour combiner des images capturées avec des données synthétiques [ALCS03], et sont intégrées dans la plupart des logiciels de modélisation et/ou de rendu.

Nous pensons que, grâce à cette adaptabilité, les cartes d'environnement peuvent être utilisées dans un contexte de rendu non-photoréaliste (Non-Photorealistic Rendering ou NPR) pour les mêmes raisons : elles représentent une solution basée-image simple et efficace. Elles permettent en particulier de combiner une stylisation 2D en précalcul sur la carte d'envirionement avec une stylisation 3D interactive une fois la carte appliquée à un objet. Néanmoins, cette double stylisation peut provoquer des problèmes de cohérences et une des contributions de cet article est de les résoudre.

De plus, l'interêt croissant apporté aux cartes d'environnement est combiné avec le développement de techniques utilisant des images à grande dynamique (High-Dynamic Range Imaging - HDRI) [RWPD06]. L'HDRI fournit un espace de travail plus précis et une intégration des données synthétiques plus facile et plus cohérente avec les images 2D. Même si ces qualités sont particulièrement avantageuses pour les rendus réalistes, elle peuvent aussi être très utiles pour les rendus expressifs. Par exemple, les images HDR peuvent permettre des segmentations plus précises, souvent utilisées dans les processus de stylisation. Les intensités des images HDR étant proches de celles de la réalité, il peut être plus facile de simuler le processus de création et de stylisation d'un artiste qui observe un environnement. Il est néanmoins nécessaire d'appliquer un algorithme de réduction de ton (tone-mapping) lorsque l'on veut afficher de telles images, et celui-ci doit être parfaitement intégré dans le processus de rendu. L'utilisation de l'HDR pour le NPR est ainsi un autre objectif de cet article.

Les principales contributions de cet article sont les suivantes : (i) Un pipeline de rendu pour faire du rendu expressif sur des objets éclairés par une carte d'environnement.



(a) (b) (c) (d)

FIG. 1: *Quelques exemples de rendus interactifs d'objets éclairés par des cartes d'environnement. (a) - style* cartoon*. (b) - style basé sur une détection des lignes caractéristiques. (c) - les luminances de l'environnement HDR ont été segmentées en six régions. Des canevas composés de points de densité plus ou moins grande sont utilisés et combinés avec la valeur de luminance de chacune de ces régions. En (d), une abstraction des couleurs a été mise en place pour les réflexions sur l'objet 3D, tandis qu'une simple détection des contours est utilisée pour l'arrière plan.*

(ii) Une amélioration de la stylisation des images par l'utilisation de données HDR. Nous présentons aussi une brève étude de l'influence d'une réduction de ton sur le résultat final. (iii) Une analyse de l'influence de chaque étape de rendu, illustrée par des styles spécifiques dans ce contexte.

Cet article est organisé de la manière suivante. Nous présentons brièvement les travaux précédents, en nous focalisant sur les rendus expressifs hybrides 2D/3D et sur l'utilisation de l'HDR dans les stylisations. Nous décrivons ensuite l'architecture de notre pipeline, puis nous faisons une comparaison des stylisations entre des images HDR/LDR (Low-Dynamic Range). Grâce à ces stylisations et à notre pipeline de rendu, nous montrons quelques résultats sur des scènes éclairées par des cartes d'environnement, puis nous discutons de ces travaux.

2. Travaux précédents

L'objectif de cette section n'est pas de faire un état de l'art complet du NPR. Nous nous focaliserons principalement sur deux aspects : la stylisation de données 2D et 3D combinées (nécessaire pour travailler avec des cartes d'environnement), et l'utilisation d'images HDR pour le NPR. Le lecteur peut se référer à [GG01, SS02] pour un état de l'art du NPR.

Si des styles similaires existent dans des espaces 2D et 3D, comme les rendus en demi-teintes (2D [SHS02] - 3D [FMS02, FV03]), les rendus à base de traits (2D [Her01] - 3D [Chi06]), très peu de travaux ont été fait pour combiner des données 2D et 3D dans des processus de rendus expressifs interactifs. Les travaux qui s'en rapprochent le plus proviennent des recherches concernant la réalité augmentée. Fischer et al. [FBS05] et Haller et al. [HLB05] ont introduits des stylisations pour améliorer la cohérence entre des données acquises et des données virtuelles. Fischer et Bartz [FB05] utilisent un filtre bilatéral [TM98] combiné avec une détection des contours de Canny [Can86] pour l'arrière plan, et un *toon shading* [LMHB00] combiné avec un rendu des silhouettes [Lan00] pour l'objet 3D. Concernant les rendus à base de traits, Fischer et al. [FBS05] utilisent un filtre simulant un pinceau sur l'image de la caméra et un système de particules pour l'objet 3D de manière à lui appliquer des coups de pinceau. Haller et al. [HLB05] utilisent un post-traitement sur l'image avec une segmentation des couleurs et une détection des contours. Si ces solutions peuvent être utilisées avec un placage d'environnement sur des scènes, nous pensons que des styles plus complexes et nombreux peuvent être mis en place avec une architecture de rendu plus adaptée, qui inclue l'éclairement des objets par l'environnement.

De nos jours, le placage d'environnement est fréquemment utilisé avec l'HDRI pour améliorer le réalisme, et notamment l'éclairement d'une scène. De plus, les publications récentes ont montrées qu'il était possible d'utiliser l'HDRI pour des travaux plus artistiques. Dans [LFUS06], les auteurs mettent en place une édition locale d'un opérateur d'adaptation de ton dans le but de s'adapter aux désirs de l'utilisateur. Pouvant être considérées comme réalistes, les changements de couleurs et d'intensités en font inexorablement des images non-photoréalistes. Dans ce domaine, l'HDR apporte une meilleure segmentation grâce à une amélioration du processus de controle de l'utilisateur. Les images HDR pourraient donc être utilisées pour les rendus NPR, et la grande précision qu'elles apportent pourrait définir une adaptation locale des stylisations. Pour un état de l'art plus précis concernant l'HDRI, l'auteur peut se référer au livre de Reinhard et al. [RWPD06].

3. Architecture

Notre architecture (cf. Figure 2) fournit un pipeline flexible pour du rendu NPR avec des objets sur lesquels on plaque une carte d'environnement. Le processus est basé sur les possibilités offertes par les GPU actuels de manière à subdiviser le rendu en étapes successives et plus spécialement décomposer la stylisation de l'environnement 2D et des données 3D. Cette section décrit les quatres étapes de ce pipeline et montre comment un style différent peut être calculé dans chacune d'entre elles, puis combinés dans l'image finale.

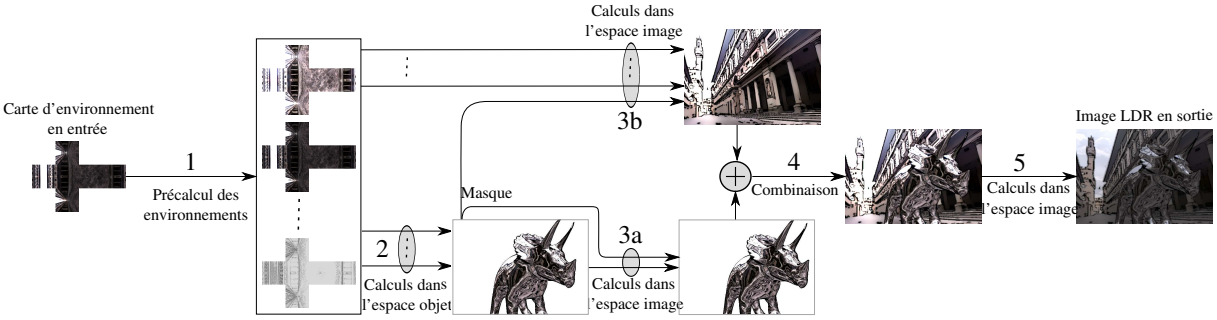


FIG. 2: *Notre pipeline de rendu pour la stylisation de scènes éclairées par des cartes d'environnement*

1. Précalcul de l'environnement : Le principal interêt de cette étape est de préparer la carte d'environnement pour le rendu interactif. Elle peut être utilisée pour calibrer les images HDR [RWPD06], pour calculer les luminances utiles à la réduction de ton, pour styliser ces environnements ou même pour générer les niveaux de détail (mipmaps). Une fois cette étape terminée, on dispose alors d'une série de *cube maps* qui seront utilisés et/ou combinés pour représenter l'environnement et pour éclairer l'objet 3D.
2. Shading dans l'espace objet : Durant l'étape suivante, les stylisations 2D sont appliquées directement sur l'objet 3D. Une ou plusieurs *cube maps* de la pile peuvent alors être utilisées et combinées. La stylisation résultante peut être augmentée ou remplacée par des styles obtenus dans l'espace objet, indépendamment de l'environnement, comme la détection des silhouettes. Chaque composant du rendu peut être modifié par l'utilisateur dans le but d'obtenir le style désiré. Cette étape est essentielle pour la cohérence du style dans la scène.
3. Calculs dans l'espace image : dans cette étape, nous fournissons des outils permettant de créer des styles différents pour l'objet et la carte d'environement afin de mieux les distinguer.
4. - 5. Combinaison et finalisation : L'objet et sa carte d'environement sont ensuites combinées pour finaliser le rendu. A ce stade, un algorithme de réduction de ton peut être mis en place, ainsi qu'une stylisation globale dans l'espace écran, comme l'ajout d'une texture de papier [CTP*03] ou un filtre (un flou comme dans [HLB05]) pour augmenter la cohérence du rendu final. Une segmentation peut aussi être utilisée pour déterminer où les différentes stylisations doivent être appliquées (cf. Figure 4).

4. Stylisation d'images HDR

Le principal objectif de cette section est de montrer au lecteur une brève vue d'ensemble de ce que peut apporter l'HDRI au NPR, et d'illustrer comment l'adaptation de ton [RWPD06] peut interagir avec le processus de stylisation. En effet, la dynamique de ces images étant supérieure à celle des images traditionnelles, il est nécessaire d'utiliser un opérateur de réduction de ton pour les afficher sur nos écrans actuels.

4.1. Adaptation de ton et stylisation

La réduction de ton peut être appliquée avant le processus de stylisation - celle-ci est alors effectuée sur des images classiques, avec une faible dynamique - ou après la stylisation. L'adaptation réduisant la dynamique des images, il se peut alors que celle-ci enlève certains détails. D'un autre coté, le processus de création d'un artiste est généralement de se placer en face de scènes réelles (i.e., données HDR réelles) et ensuite de dessiner/peindre celles-ci. C'est un peu similaire à faire une adaptation de ton après la stylisation. Dans [SK06], les auteurs l'ont expérimenté avec leur propre opérateur [SKMS06]. Pour étendre leurs travaux, nous utilisons l'opérateur de Ward [War94] avant et après trois différentes stylisations : une simple détection de contours avec le filtre laplacien d' [Ope], un style *cartoon* [WOG06] et le style pointilliste [Her98] (cf. Figure 3). Cela confirme que le fait d'appliquer la stylisation directement sur les données HDR augmente la quantité de détails détectés. Les images résultantes contiennent plus de caractéristiques (cf. Figure 3).

Il existe un grand nombre de méthodes pour adapter la dynamique [RWPD06], et chacune d'entre elles possède ses propres qualités intrinsèques. Le choix de l'opérateur d'adaptation de ton produira donc un résultat visuel différent pour une stylisation donnée. La Figure 5 montre quelques exemples. Avec la même calibration [RWPD06], sans mise à l'échelle et avec les paramètres d'adaptation de ton par défaut, l'opérateur de Ward [War94] fournit une meilleure préservation des couleurs que celui de Ferwerda [FPSG96]. Ceci est dû à l'adaptation perceptuelle de la couleur du second opérateur, qui réduit la colorimétrie des scènes très peu éclairées (comme la pénombre). De la même manière, l'opérateur local de Reinhard [RSSF02] fournit une meilleure

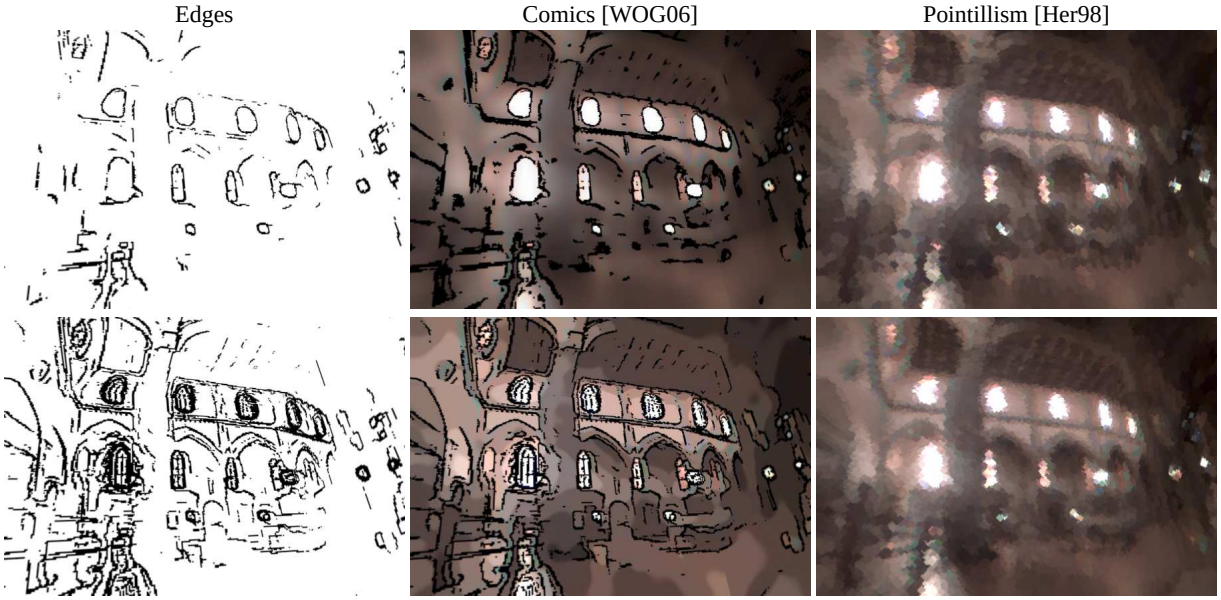


FIG. 3: *Comparaison de différents styles avec la réduction de ton appliquée avant (en haut) et après (en bas) la stylisation.*

préservation des contrastes qu'un opérateur global comme celui de Ward.

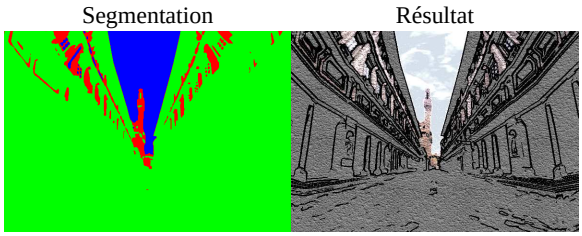


FIG. 4: *A gauche : segmentation d'une image HDR basée sur les luminances (bleu→ciel (régions de hautes intensités), rouge→régions d'intensités moyennes, vert→régions de faibles intensités). A droite : stylisation basée sur cette segmentation (bleu→papier bleu, rouge→segmentation des couleurs+canevas, vert→contours+canevas)*

4.2. Nouveaux styles avec les images HDR

Avec une grande dynamique, beaucoup plus de détails sont détectés et une segmentation plus précise peut être appliquée sur les images (comme on l'a vu dans la section précédente). Avec ces segmentations plus précises, il devient plus facile d'utiliser un style différent pour chacune des régions de la scène. Cette méthode est similaire à l'opérateur d'adaptation de ton local et paramétré par l'utilisateur développé par Lischinski et al. [LFUS06], mais appliqué dans le contexte du NPR. Dans la Figure 4, nous avons segmenté l'image en trois régions de manière très précise, avec un simple seuillage sur la luminance. Les régions de fortes intensités correpondent au ciel et les régions de faibles intensités sont celles qui sont éclairées indirectement. Un simple canevas bleu et blanc à l'aspect d'un papier est utilisé pour le ciel, une segmentation des couleurs est appliqué dans les régions intermédiaires et une détection des contours est faite dans la dernière. Le style résultant produit un bon effet de profondeur sur l'environnement, avec cette transition du ciel coloré aux contours noirs.

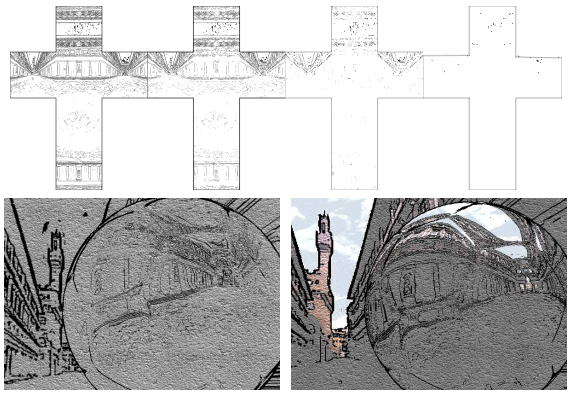
5. Stylisation d'objets avec cartes d'environnement

Pour illustrer le fonctionnement de notre pipeline de rendu, nous décrivons deux nouveaux styles. Nous exposerons alors les objectifs et les actions de chacune des étapes associées à ceux-ci.

5.1. Dessin interactif de lignes

Durant les précalculs, nous commençons par extraire les contours du *cube map* en utilisant un simple filtre laplacien [Ope]. Nous créons alors les différents niveaux de détail, comme dans [KLK*00], avec une fonction spécifique qui assure une abstraction correcte des lignes durant les transitions. Nous voulons empêcher que trop de détails apparaissent lorsque les niveaux de résolution sont faibles. Nous avons choisis une solution simple qui consiste à créer un pixel noir si trois des quatre pixels de la résolution supérieure sont noirs, et de créer un pixel blanc dans le cas contraire. La solution de simplification progressive de lignes décrite dans

FIG. 5: *Comparaison de quelques opérateurs d'adaptation de ton sur un style* cartoon *(en haut) et sur un style impressionniste (en bas). La même calibration, sans mise à l'échelle, a été utilisée pour toutes les images.*



Simple stylisation
Contours+canevas

Segmentation de la Figure 4
Contours+canevas+couleur+ciel

FIG. 6: *Style basé sur une détection des contours. Les images du haut représentent les différents niveaux de détail des lignes. L'image en bas à gauche est un rendu utilisant un canevas. Il est possible d'améliorer ce rendu en utilisant la segmentation de la Figure 4 (image en bas à droite).*

[BTS05] donnerait certainement de meilleurs résultats. Les niveaux de détail sont présentés dans la Figure 6.

Dans l'espace objet, nous combinons les réflexions du *cube map* stylisé avec une détection des silhouettes des données synthétiques en temps réel, afin d'augmenter la cohérence. Les niveaux de détail corrects sont sélectionnés durant le calcul des réflexions du *cube map*, de manière à préserver la densité des lignes sur l'écran. La détection des contours de l'objet 3D peut facilement être mise en place dans un *fragment shader*, en utilisant les fonctions prédéfinies d'OpenGL GLSL. L'algorithme suivant montre le code correspondant :

```
float d = ratio*length(fwidth(unorm));
float l = cst*(1+dot(view,normal));
if(abs(dot(view,normal)) < d)
 // contour détecté
 gl_FragColor = vec4(0.0,0.0,0.0,1.0);
else
  // pas de contours
 gl_FragColor = textureCubeLod(...,...,l);
```

normal et *view* correspondent respectivement à la normale et au vecteur de vue projetés puis normalisés dans l'espace image. *unorm* correspond à la normale non-normalisée. Le *ratio* est un paramètre choisi par l'utilisateur qui contrôle la largeur des contours. Notez que nous avons implémenté notre propre sélection de niveaux de détail (*l* dans le code) de façon à forcer le choix d'un niveau moins détaillé lorsqu'un polygone devient tangent au point de vue. Cette sélection de niveau est controllée par la constante *cst*.

L'étape d'assemblage combine simplement l'environnement et l'objet stylisé. Pour améliorer la cohérence, un canevas de type papier est multiplié avec l'image combinée lors de la dernière étape. Nous utilisons actuellement une méthode simple pour déterminer les coordonnées de texture. Elle consiste à choisir l'intersection entre le vecteur de vue et le *cube map* pour obtenir les coordonnées du centre de la texture. Néanmoins, cette solution n'évite pas complètement

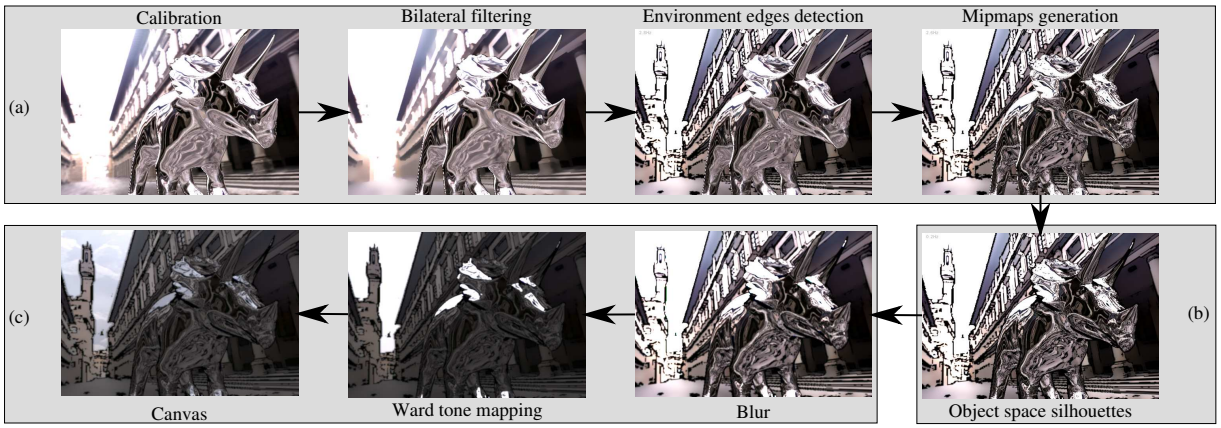


FIG. 7: *Pipeline utilisé pour créer le style* cartoon *: (a) - précalculs, (b) - shading dans l'espace objet, (c) - post-traitements. Nous montrons l'influence de chacune des étapes pour le rendu de ce style.*

l'effet "rideau de douche" produit par le placage du canevas. Les solutions décrites dans [CTP*03, BSM*07] pourraient améliorer la sensation de mouvement pendant le déplacement de la caméra. On peut voir le résultat final sur la Figure 6 (en bas à gauche).

Dans la Figure 6, l'image en bas à droite est une simple extension de ce style. Nous avons simplement décidé de segmenter l'image en trois régions avec un *fragment shader* (voir Figure 6), et d'appliquer un différent style/canevas sur chacune d'entre elles (cf. Figure 4). Pour les instensités les plus grandes (i.e., le ciel), nous utilisons un canevas représentant un papier bleu grossièrement froissé. Une simple abstraction des couleurs est utilisée pour le reste de la scène.

5.2. Style *cartoon*

Ce second style utilise plus d'étapes dans le pipeline, comme le montre la Figure 7. Il est basé sur le style présenté dans [WOG06] (pour l'abstraction de l'environnement).

Lors des précalculs, nous calibrons la carte d'environnement HDR originale [RWPD06] pour calculer un facteur global de mise à l'échelle qui sera utilisé dans les post-traitements pour l'adaptation de ton. Nous appliquons ensuite un filtre bilatéral [TM98] et une détection des contours laplacienne sur le *cube map*. La même méthode présentée dans la section précédente est utilisée pour générer les niveaux de détail des contours. Une moyenne est suffisante pour calculer les niveaux sur l'image couleur obtenue après application du filtre bilatéral.

Dans ce style, le shading de l'espace objet est similaire à celui vu précédemment. Les niveaux de détail corrects sont sélectionnés et reflétés sur l'objet, puis les silhouettes 3D sont ajoutées.

Après l'étape de combinaison, le post-traitement consiste à appliquer un opérateur d'adaptation de ton, sans faire de lecture arrière (du GPU vers le CPU) pour éviter une perte des performances. Nous avons choisis l'opérateur de mise à l'échelle de Ward [War94] car il est simple et efficace. Une implémentation GPU [GWWH03] d'un opérateur local comme [RSSF02] pourrait néanmoins donner de meilleurs résultats. La principale difficulté de l'opérateur de Ward est de calculer un facteur prenant en compte l'ensemble des valeurs de l'image affiché, directement sur le GPU. Celui-ci est calculé avec un *fragment shader* qui parcours partiellement l'image (par exemple, tous les 16 pixels) de manière à produire une texture composée seulement d'un pixel et contenant le facteur de mise à l'échelle requis. Il est maintenant facile de récupérer cette valeur dans une dernière passe pour modifier la luminance de l'ensemble des pixels. L'opérateur d'adaptation de ton est donc une simple mise à l'échelle linéaire des valeurs des pixels, et prend en compte ce facteur et celui qui a été calculé lors de la calibration. Enfin, un canevas est ajouté dans le ciel à la dernière étape du pipeline.

6. Resultats et analyse

Style	S1-640x480	S2-640x480	S2-1200x800
Contours	120	100	40
Cartoon	33	30	10

TAB. 1: *Fréquence d'affichage pour les différents styles et configuration de scènes. L'objet est composé de 14064 polygones pour toutes les configurations. La configuration 2 (S2) correspond à une vue très proche de l'objet et la configuration 1 (S1) à une vue éloignée. (voir Figure 8).*

Les féquences d'affichage ont été testées sur un PC disposant d'un processeur AMD Turion 64 $\times$2 1.6 GHz et d'une carte graphique NVIDIA Geforce Go 7600.

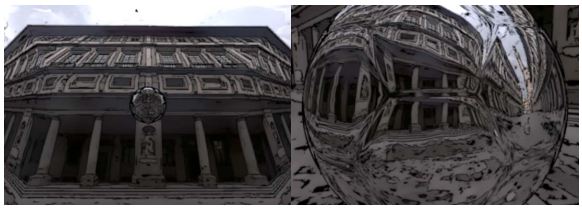


FIG. 8: *A gauche : Configuration S1 - Vue large de la scène. A droite : Configuration S2 - Vue réduite.*

Comme le montre le tableau 1, le temps de rendu est largement dépendant de la résolution de l'écran. La plupart des opérations de ces deux rendus étant faites dans le *fragment shader*, ce résultat est cohérent. De plus, on remarque le coût des post-traitements de la dernière étape du style *cartoon*. Pour le style détectant les contours, les calculs dans le *fragment shader* sont seulement faits sur les parties de l'image occupées par l'objet et il n'y a pas de post-traitements. Concernant le style *cartoon*, le post-traitement final est appliqué sur l'intégralité de l'image et dépend donc plus de la résolution. Notez que, pour ces deux styles, la résolution du *cube map* est de 6x512x512, et le temps de précalcul est environ de 20 secondes.

Nous nous sommes jusqu'ici focalisés sur la présentation de deux styles de rendu dont l'image finale est très cohérente (sensation qu'un seul style a été appliqué sur l'ensemble de l'image). Une plus grande variété de styles peuvent être mis en place comme on peut le voir dans la Figure 1, avec par exemple, des rendus différents pour l'objet et l'arrière plan. C'est le cas dans la dernière image : une abstraction des couleurs est utilisée pour l'objet et une détection des contours est faite sur l'arrière plan.

7. Conclusion

Dans cet article, nous avons introduits un pipeline flexible pour faire des rendus interactifs et stylisés d'objets sur lesquels on a plaqué un environnement à grande dynamique, combinant des stylisation 2D (sur la carte d'environnement et sur les images) et des stylisations 3D (sur l'objet). Chacune des étapes du pipeline peut facilement être modifiée par un utilisateur pour obtenir l'effet désiré. Nous avons ensuite introduits deux styles de rendu interactif basés sur ce pipeline : le premier est un simple style de détection de contours, et le second combine une détection des contours avec une abstraction des couleurs afin de produire un style de type *cartoon*. De plus, la stylisation d'images HDR apporte une qualité, une meilleure segmentation et extraction de détails, et permet d'élargir la gamme de styles NPR. Avec les styles présentés, nous avons montré que nous pouvons fournir un rendu cohérent (pour la lumière et dans l'espace image), même lorsqu'on sépare le processus pour l'objet et l'environnement.

Comme travaux futurs, nous aimerions améliorer d'une part l'interface pour obtenir un processus plus facile à utiliser. Pour le moment, tous les styles doivent être programmés. Le plus grand challenge, et le but de ce projet, est l'intégration d'objets synthétiques avec une capture de l'environnement en temps réel. Il faudra donc optimiser les précalculs.

References

[ALCS03] AGUSANTO K., LI L., CHUANGUI Z., SING N. W. : Photorealistic rendering for augmented reality using environment illumination. In *ISMAR '03 : Proc. IEEE & ACM International Symposium on Mixed and Augmented Reality* (2003), p. 208.

[BSM*07] BRESLAV S., SZERSZEN K., MARKOSIAN L., BARLA P., THOLLOT J. : Dynamic 2d patterns for shading 3d scenes. *ACM Transaction on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (2007).

[BTS05] BARLA P., THOLLOT J., SILLION F. : Geometric clustering for line drawing simplification. In *Proc. Eurographics Symposium on Rendering* (2005).

[Can86] CANNY J. : A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 6 (1986), 679–698.

[Chi06] CHI M.-T. : Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy. *IEEE Trans. Visualization and Computer Graphics 12*, 1 (2006), 61–72.

[CTP*03] CUNZI M., THOLLOT J., PARIS S., DEBUNNE G., GASCUEL J.-D., DURAND F. : Dynamic canvas for immersive non-photorealistic walkthroughs. In *Proc. Graphics Interface* (june 2003).

[FB05] FISCHER J., BARTZ D. : Stylized augmented reality for improved immersion. In *VR '05 : Proc. IEEE Conference on Virtual Reality* (2005), pp. 195–202, 325.

[FBS05] FISCHER J., BARTZ D., STRASSER W. : Artistic reality : fast brush stroke stylization for augmented reality. In *VRST '05 : Proc. ACM symposium on Virtual reality software and technology* (2005), pp. 155–158.

[FMS02] FREUDENBERG B., MASUCH M., STROTHOTTE T. : Real-time halftoning : a primitive for non-photorealistic shading. In *EGRW '02 : Proc. Eurographics workshop on Rendering* (2002), pp. 227–232.

[FPSG96] FERWERDA J. A., PATTANAIK S. N., SHIRLEY P., GREENBERG D. P. : A model of visual adaptation for realistic image synthesis. In *ACM SIGGGRAPH '96* (1996), pp. 249–258.

[FV03] FUNG J., VERYOVKA O. : Pen-and-ink textures for real-time rendering. In *Proc. Graphics Interface* (2003), pp. 131–138.

[GG01] GOOCH B., GOOCH A. : *Non-Photorealistic Rendering*. A K Peters, 2001.

[GWWH03] GOODNIGHT N., WANG R., WOOLLEY C., HUMPHREYS G. : Interactive time-dependent tone mapping using programmable graphics hardware. In *EGRW '03 : Proc. Eurographics workshop on Rendering* (2003), pp. 26–37.

[Her98] HERTZMANN A. : Painterly rendering with curved brush strokes of multiple sizes. In *ACM SIGGRAPH '98* (1998), pp. 453–460.

[Her01] HERTZMANN A. : Paint by relaxation. In *CGI '01 : Computer Graphics International 2001* (Washington, DC, USA, 2001), pp. 47–54.

[HLB05] HALLER M., LANDERL F., BILLINGHURST M. : A loose and sketchy approach in a mediated reality environment. In *GRAPHITE '05 : Proc. international conference on Computer graphics & interactive techniques in Australasia and South East Asia* (2005), pp. 371–379.

[KLK*00] KLEIN A. W., LI W., KAZHDAN M. M., CORRÊA W. T., FINKELSTEIN A., FUNKHOUSER T. A. : Non-photorealistic virtual environments. In *SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 527–534.

[Lan00] LANDER J. : Under the shade of the rendering tree. *Game Developer Magazine 7*, 2 (Feb. 2000), 17–21.

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R. : Interactive local adjustment of tonal values. *ACM Trans. Graph. 25*, 3 (2006), 646–653.

[LMHB00] LAKE A., MARSHALL C., HARRIS M., BLACKSTEIN M. : Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00 : Proc. International Symposium on Non-Photorealistic Animation and Rendering* (2000), pp. 13–20.

[Ope] OPENCV : Open computer vision library. www.opencv.org.

[RSSF02] REINHARD E., STARK M., SHIRLEY P., FERWERDA J. : Photographic tone reproduction for digital images. *ACM Trans. Graph. 21*, 3 (2002), 267–276.

[RWPD06] REINHARD E., WARD G., PATTANAIK S., DEBEVEC P. : *High Dynamic Range Imaging - Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann Publishers, 2006.

[SHS02] SECORD A., HEIDRICH W., STREIT L. : Fast primitive distribution for illustration. In *EGRW '02 : Proc. Eurographics workshop on Rendering* (2002), pp. 215–226.

[SK06] SMITH K., KRAWCZYK G. : NPR for HDR, stylizing with high dynamic range photographs. In *NPAR 2006 Poster* (June 2006).

[SKMS06] SMITH K., KRAWCZYK G., MYSZKOWSKI K., SEIDEL H.-P. : Beyond tone mapping : Enhanced depiction of tone mapped HDR images. *Comp. Graph. Forum (Proc. EUROGRAPHICS 2006) 25*, 3 (Sept. 2006), 427–438.

[SKS02] SLOAN P., KAUTZ J., SNYDER J. : Precomputed radiance transfer for real-time rendering in dynamic, 2002.

[SS02] STROTHOTTE T., SCHLECHTWEG S. : *Non-Photorealistic Computer Graphics*. Morgan Kaufmann Publishers, 2002.

[TM98] TOMASI C., MANDUCHI R. : Bilateral filtering for gray and color images. In *ICCV '98 : Proc. the Sixth International Conference on Computer Vision* (1998), p. 839.

[War94] WARD G. : *Graphics gems IV*. Academic Press Professional, Inc., 1994, ch. A contrast-based scalefactor for luminance display, pp. 415–421.

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B. : Real-time video abstraction. *ACM Trans. Graph. 25*, 3 (2006), 1221–1226.

(a) (b) (c) (d)

FIG. 1: *Quelques exemples de rendus interactifs d'objets éclairés par des cartes d'environnement. (a) - style* cartoon*. (b) - style basé sur une détection des lignes caractéristiques. (c) - les luminances de l'environnement HDR ont été segmentées en six régions. Des canevas composés de points de densité plus ou moins grande sont utilisés et combinés avec la valeur de luminance de chacune de ces régions. En (d), une abstraction des couleurs a été mise en place pour les réflexions sur l'objet 3D, tandis qu'une simple détection des contours est utilisée pour l'arrière plan.*

[War94] [FPSG96] [RSSF02]

FIG. 5: *Comparaison de quelques opérateurs d'adaptation de ton sur un style* cartoon *(en haut) et sur un style impressionniste (en bas). La même calibration, sans mise à l'échelle, a été utilisée pour toutes les images.*

Segmentation Résultat

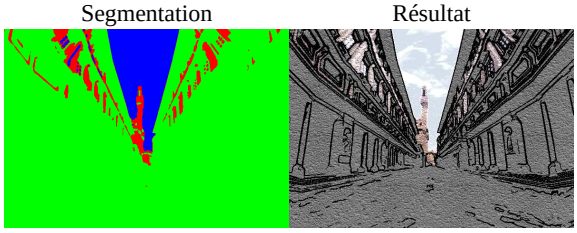
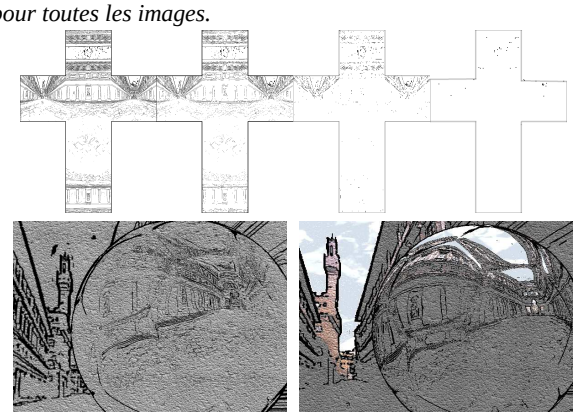


FIG. 4: *A gauche : segmentation d'une image HDR basée sur les luminances (bleu→ciel (régions de hautes intensités), rouge→régions d'intensités moyennes, vert→régions de faibles intensités). A droite : stylisation basée sur cette segmentation (bleu→papier bleu, rouge→segmentation des couleurs+canevas, vert→contours+canevas)*



Simple stylisation Contours+canevas

Segmentation de la Figure 4 Contours+canevas+couleur+ciel

FIG. 6: *Style basé sur une détection des contours. Les images du haut représentent les différents niveaux de détail des lignes. L'image en bas à gauche est un rendu utilisant un canevas. Il est possible d'améliorer ce rendu en utilisant la segmentation de la Figure 4 (image en bas à droite).*

Chaîne de Traitement pour la Numérisation et le Rendu Réaliste de Peintures d'Art

Frédéric Larue† Lucas Ammann† Jean-Michel Dischler†

LSIIT UMR 7005 CNRS-ULP - Université Louis Pasteur Strasbourg I, France

Résumé

La numérisation et la visualisation sont aujourd'hui deux domaines d'importance dans le cadre de l'héritage culturel, pour la conception de galeries d'art virtuelles par exemple. Malgré de nombreux travaux, permettre la navigation temps réel autour de copies numériques complexes demeure, dans le cas général, un problème difficile du fait de la complexité de la mesure et de la quantité de données à traiter. Dans ce papier, nous introduisons une nouvelle chaîne de traitement dédiée autant à la numérisation qu'au rendu réaliste de peintures d'art. Nous exploitons le fait que les variations géométriques pour de tels objets, même si elle sont généralement faibles, restent non négligeables d'un point de vue visuel. Contrairement à la plupart des méthodes existantes, nous proposons donc d'acquérir la géométrie en plus de la texture couleur. Ces deux informations sont exploitées lors du rendu en utilisant, pour la couleur, un modèle analytique estimé à partir des mesures et, pour la géométrie, une approche hybride qui combine deux techniques de rendu de relief selon l'échelle, fournissant ainsi un schéma adaptatif permettant un rendu temps réel, quelque soit le point de vue. Notre protocole garantit donc un rendu préservant le relief de la toile, ainsi qu'une représentation très compacte de la copie numérique finale.

1. Introduction

Dans le domaine de l'informatique graphique, de nombreux travaux s'évertuent, depuis de nombreuses années, à améliorer la qualité des images de synthèse grâce à des objets de plus en plus complexes, comme ceux produits par les dispositifs de numérisation par exemple. Mais la qualité visuelle n'est pas le seul objectif : la vitesse des algorithmes de rendu est également un élément d'intérêt en informatique graphique, et de nombreux chercheurs tentent de trouver des compromis acceptables entre vitesse et qualité visuelle.

Dans le domaine de l'héritage culturel, l'acquisition numérique est désormais couramment utilisée pour la conservation et la diffusion de pièces d'art. Le projet Michelangelo [LPC*00], par exemple, s'est fixé pour objectif de numériser les principales sculptures de Michel-Ange, avec pour résultat une base de données de maillages 3D très détaillés. En effet, les scanners 3D fournissent aujourd'hui des données qui sont de plus en plus précises, mais donc forcément de plus en plus complexes à manipuler. Du fait de la grande variété d'objets susceptibles d'être numérisés, il est intéressant de se poser la question de l'usage de méthodes d'acquisition et de rendu dédiées à certaines classes d'objets. Dans cet article, nous nous sommes concentrés sur la numérisation et le rendu réaliste temps réel de peintures d'art. Nous souhaitons offrir une methode simple d'un point de vue technique mais qui permette la conception de galerie d'art numérique, c'est à dire en préservant une certaine précision visuelle.

L'état de l'art sur la numérisation de tableaux reste assez pauvre. De manière générale, la géométrie est approximée par un simple plan et l'information chromatique capturée par un ensemble de photographies. Cependant, dans la réalité, la surface d'une toile peinte peut présenter de nombreuses aspérités, dont certaines sont suffisament significatives pour devoir être prises en compte lors du rendu si l'on souhaite garantir un certain réalisme. En fait, le relief joue un rôle important, et les effets qu'il induit sur la toile sont même parfois souhaités par le peintre lui même, comme pour la peinture au couteau par exemple. Ainsi, l'information geometrique doit être considérée autant lors de l'acquisition que pour ce qui est du rendu.

† e-mail: {larue, ammann, dischler}@lsiit.u-strasbg.fr

Figure 1: Gauche*: photographie du Tableau du Port.* Milieu*: rendu par plaquage de texture classique sur un simple plan.* Droite*: rendu à l'aide notre méthode hybride, adjoint d'une texture bidirectionnelle.*

Nous proposons une nouvelle méthode pour acquérir précisément les informations géométriques et chromatiques d'une peinture d'art, ainsi qu'un algorithme pour la visualisation des copies numériques obtenues. Nous utilisons un modèle analytique pour représenter la texture de manière réaliste et une approche hybride pour afficher les détails géométriques, accélérée par un schémas adaptatif. Nous montrerons que le rendu temps réel est atteint et que les données représentant la copie finale sont particulièrement compactes. Nos principales contributions sont donc :

- Une chaîne de traitement, allant de l'acquisition du relief et de la texture jusqu'à leur rendu final,
- Un jeu de pré-traitements spécifiquement dédiés aux données provenant de la numérisation de tableaux,
- Un algorithme efficace permettant, pour un faible coût mémoire, d'obtenir un rendu réaliste en temps réel grâce à un mécanisme adaptatif spécifique.

Après une brève discussion concernant les travaux antérieurs, nous présentons le principe de notre chaîne de traitement (section 3), incluant la numérisation et l'algorithme de rendu, respectivement décrits dans les sections 4 et 5. Ensuite, nous présentons quelques résultats en section 6 et concluons avec des perspectives possibles pour ces travaux en section 7.

2. Travaux antérieurs

Concernant la numérisation de tableaux d'art, nous pouvons citer Tominaga *et al.* [TTK04], qui ont développé une technique permettant l'acquisition de la couleur d'une peinture à l'aide d'une caméra multi-bande. Dans ces travaux, le relief de la surface n'est considéré qu'au moment de l'acquisition : la rugosité n'est utilisée que pour déterminer la reflectance spectrale de la peinture et non durant la phase de rendu. Les travaux de Grattoni *et al.* [GS03] présentent une nouvelle méthode pour numériser des surfaces peintes, en termes de géométrie et de couleur, à l'aide d'un outil optique. Cependant, la géométrie considérée est celle du support de la peinture (et non celle de la peinture elle même) puisqu'il s'agit ici de numériser des surfaces telles que des fresques dans un soucis de conservation et de restauration. Par rapport à la restauration, les auteurs de [GAL03] montrent comment ils numérisent à l'aide d'une caméra 3D une peinture sur bois de Léonard De Vinci. Ce projet utilise la géométrie pour fournir un diagnostique sur l'état du support en bois de la peinture, mais ne considère pas le relief à des fins de visualisation, contrairement à notre approche.

Pour produire un rendu réaliste de surface complexe présentant un relief de faible amplitude, il existe de nombreuses solutions. La première est, bien évidemment, l'utilisation d'un maillage extrêmement détaillé, mais cette solution est généralement peu attrayante pour le rendu temps réel. Une seconde peut être la simulation visuelle de ce relief, qui fournit de très bonnes performances d'affichage par rapport à l'utilisation de maillages. Le *bump mapping* [Bli78] a été la première méthode de ce genre. Un plaquage de texture modifié est utilisé pour perturber les normales à la surface de l'objet, simulant le relief par les modifications induites sur le calcul d'illumination. Les résultats visuels sont bons, les performances excellentes, et certains travaux ont déja proposé de faire l'acquisition de textures *bump* à partir d'objets réels [RTG97]. Malheureusement, certains problèmes de parallax peuvent survenir pour les angles d'observation rasants, d'autant plus lorsque le relief à simuler est important. Des techniques comme le *displacement mapping* [Coo84] ont alors proposé de supprimer ces problèmes en modifiant directement la géométrie sous jacente. Plus récemment, le *relief map-*

ping [OBM00,POC05,BD06] et le *parallax mapping* [Tat06] ont été introduits pour ajouter de manière purement visuelle des détails géométriques à l'aide de techniques complexes de plaquage de textures, ne nécessitant que des objets 3D basiques comme support de rendu, et fournissant donc des rendus rapides et de relativement bonne qualité.

En plus de la géométrie, l'autre caractéristique importante d'une peinture est, bien entendu, sa couleur, qui doit également être acquise très précisémment afin d'être restituée plus tard de manière fidèle. Les changements de couleur sur la surface sont liés à la variation spatiale de la réflectance induite par les différentes peintures utilisées. De nombreux travaux se sont attachés à la mesure et au rendu de réflectances de surfaces. La *fonction de texture bidirectionnelle* (*BTF*) [DNGK97] a été introduite dans ce but. Une BTF encode les variations spatiales d'un matériau, incluant les changements de BRDF, sur la surface d'un objet. Comme la BTF est une fonction à six dimensions, peu attrayante pour le rendu temps réel dû à sa grande consommation mémoire, des méthodes ont cherché à compresser les données ainsi acquises [LFTG97, MWL*99, NDM05]. Pour plus d'informations, un survol des BTF a été publié par Müller *et al.* [MMS*05]. A l'instar de McAllister *et al.* [MLH02], notre méthode utilise un modèle de Lafortune [LFTG97] pour représenter la réflectance en chaque point de la surface du tableau. Comme souligné dans [MMK03], la mesure de BTF pour des surfaces à relief important introduit de grandes imprécisions. C'est pourquoi notre méthode se concentre à la fois sur l'acquisition et le rendu du relief de la peinture au lieu d'utiliser directement une BTF.

La plupart des techniques précédemment citées sont, sous certaines conditions, peu adaptées du fait de leur coût en temps de calculs ou de leur qualité de rendu inadéquate. Les méthodes hybrides peuvent alors fournir une alternative intéressante. De tels mécanismes consistent à utiliser des algorithmes de rendu différents en fonction de certains paramètres, comme la position relative du point de vue ou la taille de l'objet projeté à l'écran. Becker *et al.* [BM93], par exemple, utilisent alternativement trois algorithmes différents, selon le point de vue et sa distance à la scène. Le principale problème des methodes hybrides résident dans les transitions entre les différents modes de rendu, qui sont souvent enclines à générer des artefacts visuels. Comme le souligne Heidrich *et al.* [HDKS00], les artefacts les plus importants proviennent des différences d'illumination entre les differents algorithmes. Généralement, une composition alpha (parfois améliorée, comme dans [GW07]) permet d'éviter ces effets de 'popping' en proposant des transitions progressives. Ces problèmes mis à part, les méthodes hybrides restent des outils puissants pour accélérer le processus de rendu. Cependant, il n'existe pas de solutions générales : chaque technique est adaptée à une situation spécifique. C'est pourquoi nous avons choisi d'introduire notre propre méthode, spécifiquement conçue pour le rendu de peintures numérisées.

3. Principe

La méthode proposée ici est basée sur le fait que la plupart des peintures d'art présentent souvent des caractéristiques géométriques qui méritent d'être préservées. Une partie du relief au dessus de la toile est, en effet, généralement significative d'un point de vue visuel. Mais la grande majorité du relief reste néammoins négligeable et peut donc être approximée par une technique de rendu simpliste. Nous proposons alors de découpler le relief significatif du reste.

Tout d'abord, la surface du tableau est approximée par un champ de hauteurs, c'est à dire une carte définissant en chaque point une élévation par rapport au plan de la toile. Cette représentation est particulièrement adaptée aux tableaux, du fait de la forme hautement planaire de ce type d'objets. Le relief est alors classifié par seuillage du champ de hauteurs. Les points en dessous du seuil sont affichés à l'aide d'un simple *bump mapping* et les autres, assimilés au relief significatif, sont représentés par un ensemble de boîtes situées au dessus du plan de la toile et affichées à l'aide d'un rendu de reliefs plus complexe. La fusion des deux techniques est gérée par un méchanisme adaptatif, accélérant de manière drastique la vitesse de rendu.

Le rendu couleur du tableau est finalement obtenu grâce à une description des variations spatiales de son matériau, représentée par une texture de lobes de Lafortune [LFTG97] estimés à partir des mesures et évalués à la volée par le matériel graphique.

4. Acquisition et Pré-Traitements

Dans cette section, nous explicitons nos choix concernant l'acquisition de la géométrie et de la texture. Nous présentons également les pré-traitements nécéssaires à notre algorithme hybride de rendu de peintures numérisées.

4.1. Acquisition 3D

La géométrie est acquise à l'aide d'un scanner à lumière structurée. Etant donné que les détails chromatiques peuvent présenter de forts contrastes, nous capturons plusieurs cartes 3D, avec différents temps d'exposition, afin de garantir la bonne acquisition des régions claires et sombres. Du fait de la forme particulière de la classe d'objets que nous nous proposons de numériser, un seul point de vue est utilisé pour l'acquisition de l'information 3D, choisi perpendiculairement au plan de la toile. La paramétrisation 2D de la carte 3D et l'angle de vue quasi-orthogonal nous fournissent une représentation géométrique déjà proche du champ de hauteurs que nous désirons obtenir. Voyons comment extraire proprement ce dernier.

4.2. Extraction automatique de la toile

Selon l'application, on peut vouloir supprimer l'information qui ne concerne pas directement la surface peinte. Il est alors

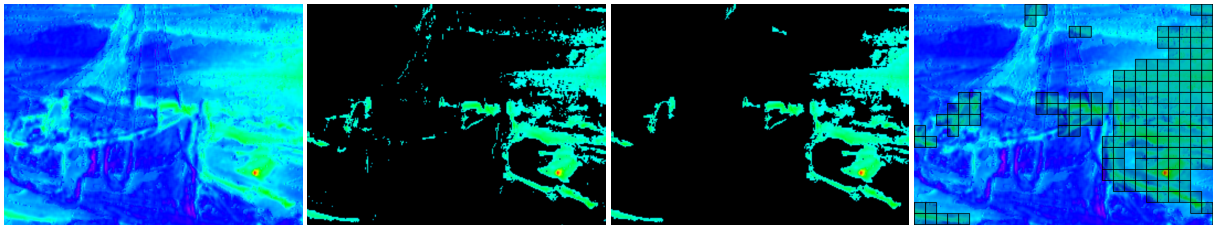


Figure 2: *Classification du relief. De gauche à droite:* ***1)*** *champ de hauteurs résultant de l'alignement par ACP de la carte de profondeurs acquise, mise en valeur par un gradient de couleur,* ***2)*** *seuillage des coordonnées de hauteurs,* ***3)*** *nettoyage pour réduire l'influence du bruit,* ***4)*** *boîtes créées autour des composantes connexes restantes et utilisées plus tard pour le rendu du relief significatif.*

intéressant de pouvoir disposer d'un traitement permettant d'extraire la toile en ignorant le cadre et toute autre information géométrique non pertinente. La particularité d'un tableau numérisé est la forme hautement planaire de sa géométrie. En effet, son relief peut être vu comme une variation de hauteur par rapport à un plan de référence. Pour récupérer le champ de hauteurs correspondant au tableau, nous effectuons une analyse en composantes principales (ACP) de la carte 3D afin d'extraire un nouveau repère local $(\vec{X},\vec{Y},\vec{Z})$, où $\vec{X}$ et $\vec{Y}$ sont les axes de principale dispersion, tangeant au plan de la toile, et $\vec{Z}$ est l'axe orthogonal. Les positions des points de la carte 3D sont alors recalculées par rapport à ce nouveau repère, et la coordonnée associée à l'axe $\vec{Z}$ est appelée *coordonnée de hauteur*. En considérant ce nouveau repère, la transition entre la toile et le cadre provoque nécessairement une discontinuité importante de la hauteur comparé aux petits détails géométriques de la toile elle même. Ainsi, nous appliquons l'opérateur de Sobel sur les coordonnées de hauteur afin de mettre en valeur ces fortes discontinuités. Pour détecter la partie intérieure de la peinture (la toile), nous appliquons un algorithme de croissance de région qui s'arrête lorsqu'une zone de gradient trop important est rencontrée. Nous choisissons comme point de départ de la croissance le centre de la carte 3D, puisqu'il est évident qu'un tableau correctement mesuré se trouve bien centré dans le champs du dispositif d'acquisition. Seule la région résultant de ce remplissage est conservée, ignorant ainsi les points qui n'appartiennent pas à la toile. Une érosion de quelques pixels est ensuite appliquée afin d'éviter la présence d'artefacts résiduels. Une fois que seule l'information pertinente demeure, une nouvelle ACP est appliquée pour recalculer un repère local $(\vec{X},\vec{Y},\vec{Z})$ qui soit plus représentatif des seules données de la toile.

4.3. Classification du relief

Notre but est maintenant d'extraire la partie du relief de la toile qui est suffisamment significative, d'un point de vue visuel. Nous définissons un seuil δ_0 en dessous duquel la hauteur est considérée comme négligeable. Actuellement, ce seuil est fourni manuellement. Durant nos tests, nous avons fixé sa valeur entre 20% et 40% de la hauteur maximale de la toile. Tous les pixels de la carte 3D dont la hauteur est supérieure à δ_0 sont marqués *valides*, et les autres *non valides*, de manière à construire un masque binaire $\mathcal{M}_v$. Les composantes connexes de $\mathcal{M}_v$ qui sont trop petites (quelques pixels) sont supprimées de manière à ignorer la possible influence du bruit de numérisation.

La grille de pixels de la carte 3D est ensuite subdivisée en un ensemble de cellules de $n \times n$ pixels. Nous verrons plus tard, dans la section 5, que la valeur de n est importante pour notre algorithme de rendu hybride puisqu'il permet de contrôler la granularité du mécanisme adaptatif. Les cellules qui ne contiennent aucun pixel valide dans $\mathcal{M}_v$ sont supprimées. Pour les autres, une boîte est créée, dont les dimensions sont définies par la taille n en pixel de la cellule, et par la hauteur maximale de la toile. Le relief segmenté est alors totalement inclus dans l'ensemble final de boîtes, comme l'illustre la figure 2. Cet ensemble sera utilisé pour rendre séparément la partie significative du relief. De plus, nous calculons également pour chaque boîte la hauteur h_{max} et les coordonnées image (u_{max}, v_{max}) de son point le plus élevé, ce qui nous sera utile pour le rendu adaptatif, décrit plus tard en section 5.3. Il est évident, mais important, de noter que plus n est petit, plus il y aura de boîtes qui seront créées. Le choix de n est discuté dans la section résultats (section 6).

4.4. Acquisition de la texture bi-directionnelle

Concernant l'information chromatique, l'apparence de la toile est capturée à l'aide de plusieurs photographies, prises pour des points de vue et des éclairages différents. La localisation de la caméra est effectuée à l'aide de la paramétrisation par lumière structurée décrite dans [LD06], et la localisation de la source lumineuse est obtenue par une structure mécanique rigide permettant de couvrir l'ensemble de l'hémisphère au dessus de l'objet.

Une fois les photographies recalées sur la géométrie, des échantillons de luminance sont extraits pour chaque point de la carte 3D par projection dans l'espace image de chaque photographie. Ces échantillons sont composés des directions locales d'observation et d'illumination et de la couleur du pixel atteint. Parmis ces échantillons, certains sont sujets aux

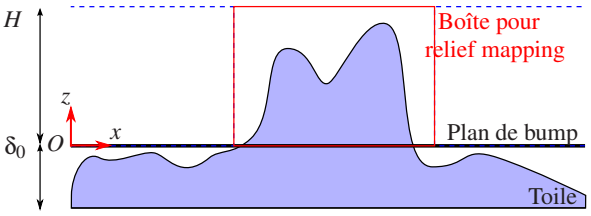


Figure 3: *Positions relatives du plan de bump et des boîtes. La boîte rouge contient une partie du relief significatif.*

auto-occlusions, c'est à dire lorsque la lumière est stoppée par la géométrie avant d'arriver en ce point. Certains modèles analytiques de représentation de texture sont sensibles, en termes de précision, aux fortes discontinuités que peuvent représenter de tels effets d'ombre. Comme l'information de relief est connue dans notre cas, nous choisissons d'éliminer les échantillons qui ne correspondent pas à de l'illumination directe. Ceux-ci sont détectés par lancé de rayons.

Un modèle de Lafortune est finalement estimé en chaque point de la carte 3D. Nous utilisons l'approximation de [MLH02] à un lobe spéculaire, comme dans l'équation 1 :

$$f_r(\vec{v},\vec{l}) = \rho_d + \rho_s \left(C_x v_x l_x + C_y v_y l_y + C_z v_z l_z\right)^k \qquad (1)$$

où ρ_d et ρ_s sont des vecteurs RVB décrivant respectivement les contributions diffuses et spéculaires à l'énergie lumineuse sortante, et $\vec{v}$ et $\vec{l}$ sont les directions locales d'observation et d'incidence lumineuse. Nous avons choisi ce modèle pour sa simplicité, permettant une évaluation directe par le matériel graphique, et pour sa compacité, étant donné que seules trois textures 2D sont nécessaires : deux pour ρ_d et ρ_s et une pour les paramètres (C_x, C_y, C_z, k) de forme du lobe. Evidemment, d'autres modèles de représentation peuvent être utilisés, mais nous rappelons que l'un de nos objectifs est la compacité de la copie numérique finale.

5. Rendu

Après classification du relief, nous obtenons deux sortes de relief, le *négligeable* et le *significatif*, chacun étant affiché à l'aide d'une technique différente : le *bump mapping* et le *relief mapping*, respectivement. Ces deux techniques sont fusionnées grâce à un mécanisme qui choisit automatiquement le meilleur algorithme selon les conditions d'observation, accélérant ainsi le rendu. Le relief significatif est représenté par l'ensemble de boîtes déterminé au cours des traitements précédents. En plus de ces boîtes, nous créons une texture contenant le champ de hauteurs défini précédemment ainsi que les vecteurs normaux en chaque point de la toile. Ces normales sont calculées en considérant l'ensemble des triangles qu'il est possible de construire avec le 8-voisinage d'un point dans la carte 3D. Seules les composantes n_x et n_y sont stockées, la troisième étant recalculée à la volée. Cette texture est utilisée pour le rendu des deux types de relief.

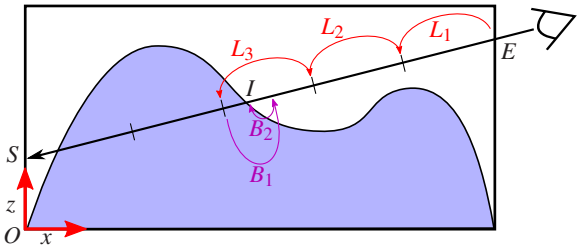


Figure 4: *Recherche d'intersection entre un rayon et un champ de hauteurs effectuée sur GPU par l'algorithme du* relief mapping.

5.1. Rendu du relief négligeable

Pour le relief négligeable, la surface de la toile est approximée par un plan qui est rendu en utilisant un *bump mapping* classique. Comme la partie significative du relief ne commence qu'au delà du seuil δ_0, un espace existe entre la hauteur minimum de la toile et la hauteur de la partie basse des boîtes. Pour palier à ce problème, nous déplaçons le plan de bump jusqu'à ce seuil. Comme l'illustre la figure 3, la vraie toile se trouve en fait sous le plan de bump, mais la hauteur maximale de la toile reste la même. Le relief de la peinture reste ainsi totalement respecté.

5.2. Rendu du relief significatif

L'algorithme de rendu utilisé pour le relief significatif est une variante de l'algorithme du *relief mapping* décrit par Policarpo *et al.* [POC05]. Les zones rendues à l'aide de cet algorithme sont superposées au plan de bump, ce qui permet de corriger les erreurs de parallax introduites par ce dernier lorsque le relief devient trop important.

Rappel sur le relief mapping Le *relief mapping* est une extension du plaquage de textures classique. Il utilise la GPU pour modifier la géométrie de l'objet rendu. Pour chaque pixel rendu d'un polygone simple, un rayon est lancé pour déterminer une intersection avec la surface réelle de l'objet, qui est définie par un champ de hauteurs dont l'intensité des pixels est interpretée comme une variation de hauteur au lieu d'une simple couleur. Cette intersection est trouvée par une recherche linéaire, suivie d'un raffinement binaire, respectivement représentés par les étapes L_i et B_j de la figure 4. Lorsque l'intersection entre le rayon et la surface réelle est trouvée, l'illumination du pixel peut être calculée.

Relief mapping à partir de boîtes Habituellement, l'algorithme du *relief mapping* utilise un simple polygone comme support de rendu. Notre méthode utilise des boîtes définies dans l'espace 3D, chacune étant définie par sa position (u_b, v_b) et sa taille (w_b, h_b) dans l'espace texture du champ de hauteurs. Chaque boîte est donc associée à une portion de la texture, qui est, par conséquent, partagée entre toutes les boîtes. Les hauteurs minimales et maximales sont

Figure 5: *Comportement de notre rendu adaptatif. Pour les gros plans (gauche) ou les angles rasants (droite), un nombre plus important de boîtes est automatiquement sélectionné afin de compenser l'erreur de parallax introduite par le bump.*

les mêmes pour toutes les boîtes, et correspondent respectivement au niveau du plan de bump et à la hauteur H la plus élevée de toute la toile.

Pour un rendu efficace du *relief mapping* à partir de boîtes, nous associons des coordonnées de texture 3D à chaque sommet afin de définir un repère local à chaque boîte (O_x et O_z sur la figure 4) à l'intérieur duquel le lancé de rayon sera effectué. Au cours du rendu, ces coordonnées de texture fournissent immédiatement le point d'entrée du rayon dans la boîte, quelqu'en soit la face considérée. L'algorithme de *relief mapping* procède alors ainsi : pour chaque pixel atteint par la boîte actuellement rendue, les points d'entrée E et de sortie S du rayon d'observation sont déterminés à l'aide des coordonnées de texture 3D. Ces deux points nous donne le rayon de traversée le long duquel la recherche d'intersection avec le champ de hauteurs doit être faite. Les phases de recherche linéaire et binaire sont alors effectuées de la même manière que pour le *relief mapping* standard.

5.3. Mécanisme de rendu adaptatif

Une peinture observée selon une direction proche de la normale à sa surface ne présente quasiment aucune différence visuelle entre le relief significatif et le relief négligeable. Cela est également valable lorsque le point de vue est éloigné, c'est-à-dire lorsque lorsque les variations de hauteur deviennent négligeables par rapport à leur taille projetée à l'écran. En effet, la distance entre la surface réelle et le plan de bump devient alors trop petite pour être visible.

Nous avons donc développé un algorithme pour déterminer automatiquement si une boîte doit être affichée ou non. De plus, une composition alpha est effectuée pour éviter d'éventuels artefacts de 'popping'. L'état d'une boîte est donné par un coefficient α_{box} calculé à partir du plus haut point I de la boîte, dont les coordonnées dans la scène sont facilement récupérées à partir des valeurs h_{max} and (u_{max}, v_{max}) calculées précédement (voir section 4.3). En considérant la projection B de I sur le plan de bump le long de la direction d'observation $\overrightarrow{v}$, nous pouvons voir que le ratio $r = \|\overrightarrow{IB}\| \ / \ \|\overrightarrow{CI}\|$, où C est la position de la caméra, dépend directement de l'angle de vue, mais également de la distance au point de vue, comme l'illustre la figure 6.

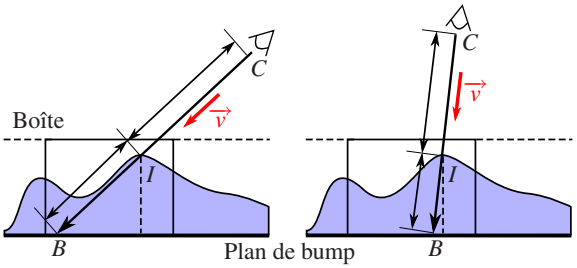


Figure 6: *Calcul du coefficient de transition α_{box} à partir du point le plus haut de chaque boîte I. Le ratio $\|\overrightarrow{IB}\|/\|\overrightarrow{CI}\|$ dépend de l'angle de vue et de la distance à la surface.*

Ce ratio tend vers zéro pour des points de vues éloignés ou frontaux, et augmente lorsque l'on se rapproche ou que l'angle de vue devient rasant. Ce ratio est donc pertinent pour notre mécanisme adaptatif. Le calcul par boîte du coefficient α_{box} est donné par l'équation 2:

$$\alpha_{box} = \begin{cases} 0 & \text{si } Kr < \varepsilon_{min} \\ 1 & \text{si } Kr > \varepsilon_{max} \\ \frac{Kr - \varepsilon_{min}}{\varepsilon_{max} - \varepsilon_{min}} & \text{sinon} \end{cases} \quad (2)$$

où ε_{min} et ε_{max} sont deux valeurs seuil et K est un facteur défini par l'utilisateur pour contrôler la sensibilité de la transition entre les deux algorithmes de rendu. Selon la valeur de α_{box}, il y a trois cas à considérer. Si α_{box} est nul, la boîte n'est pas dessinée. Si $\alpha_{box} = 1$, la boîte est affichée sans composition alpha. Si $\alpha_{box} \in]0,1[$, la boîte est composée avec le plan de bump.

Généralement, des inconsistances lumineuses peuvent apparaître lorsque différents algorithmes de rendu sont combinés. Dans notre cas, la même normale, la même information de couleur et le même modèle d'illumination sont utilisés pour les deux algorithmes. Ainsi, aucune différence d'illumination n'est visible au niveau des transitions, comme on peut le voir sur la figure 7.

5.4. Illumination

Comme décrit en section 4.4, nous utilisons un modèle de Lafortune pour calculer l'illumination. L'équation 1 est di-

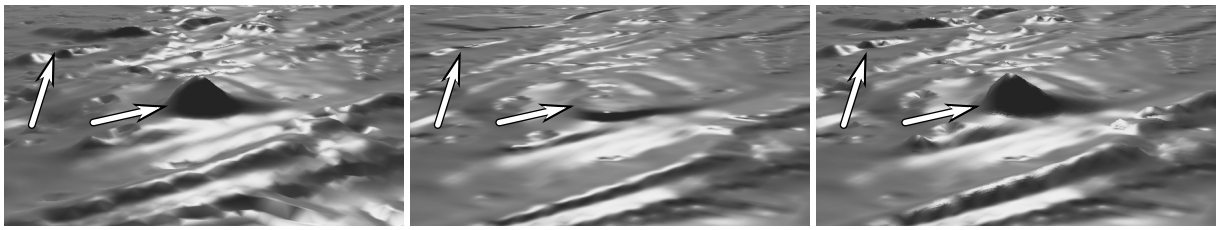


Figure 7: *Comparaison entre le rendu maillage (gauche), le* bump mapping *(milieu) et notre rendu hybride (droite). L'erreur de parallax introduite par le* bump mapping *est corrigée par l'utilisation de* relief mapping *là où le relief devient significatif.*

rectement évaluée par la carte graphique. L'information de couleur est disponible via deux textures RGB et une RGBA. Les direction d'illumination et d'observation sont toutes deux corrigée à l'aide de la coordonnée de hauteur, même pour le bump, de manière à prendre en compte la vraie position de la surface, et le vecteur normal est recalculé à la volée à partir des coordonnées n_x et n_y estimées précédemment (voir section 5).

6. Résultats et Discussions

Nous avons testé notre méthode sur le Tableau du Port, illustrée en figure 1 par une photographie originale et un rendu de notre copie numérique. Le principal avantage de notre méthode est l'absence de maillage pour représenter la géométrie, ce qui engendre un faible coût mémoire. En fait, la carte 3D capturée par le scanner est directement traitée et utilisée comme champ de hauteurs, travaillant donc directement sur l'échantillonnage gréométrique initial. Seules quatre textures flottantes 16bits (une RGBA et trois RGB) et une liste de boîtes sont nécessaires pour représenter la copie numérique finale, incluant la texture bidirectionnelle. La taille totale après traitement est d'environ 11Mo. Le maillage correspondant contient 815K triangles.

Les performances obtenues avec notre méthode hybride, reportées dans la table 1, sont temps réel. Les fréquences d'affichage ont été mesurées sur un AMD Athlon X2 4200+ doté d'une carte NVIDIA GeForce 7900GTX. On peut voir que l'accélération apportée par l'adaptativité est significative, surclassant systématiquement le *relief mapping* standard. Notre approche est cependant moins efficace qu'un rendu de maillages pour des points de vue très proches. En effet, la vitesse du rendu adaptatif dépend fortement du point de vue. A angles rasants ou en gros plans, le relief est beaucoup plus visible et l'adaptativité requière plus de boîtes, comme le montre la figure 5, diminuant d'autant les performances. Ce mécanisme dépend aussi du nombre total de boîtes, qui est directement lié à leur taille. Pour une petite taille, les boîtes approximent bien le relief mais le calcul de visibilité par boîte devient plus coûteux. Au contraire, une taille de boîte plus grande entraîne un nombre de pixels plus important à rendre par *relief mapping*, algorithme plus coûteux que le simple *bump mapping*. Des tests de performance sont donnés dans la table 2 pour différentes tailles de boîte.

Dist.	Angle de vue	Maillage	RM	Hybride	Hybride + Adapt.
proche	rasant	71	36	33-48	36-57
	face	85	32	39-48	56-72
cadré	rasant	60	100	61	330
	face	52	74	60	350
loin	rasant	96	280-450	82	650
	face	96	155-230	80	1250

Table 1: *Vitesse d'affichage (Hz) pour le Tableau du Port avec différents algorithmes de rendu et pour différentes conditions d'observation.*

Taille boîte	♯ Boîtes	Face	Interm.	Rasant
5×5	2073	305	142	18
10×10	714	300	185	38
20×20	253	330	250	75
50×50	81	240	175	145
100×100	35	135	130	125

Table 2: *Vitesse de rendu (Hz) de notre méthode hybride adaptative pour différentes tailles de boîte et différents angles d'observation.*

Le *relief mapping* est particulièrement adapté au rendu de peintures d'art du fait de leur nature planaire. En effet, les peintures traditionnelles peuvent être correctement représentées par des champs de hauteur. L'autre avantage à utiliser le *relief mapping* est la correction des erreurs de parallax introduites par le *bump mapping*, comme le montre la figure 7. Cependant, il est important de remarquer que le rendu de champs de hauteur texturés impose certaines limitations. En effet, si un saut important se produit entre les hauteurs de deux pixels adjacents, la couleur le long de la falaise générée est simplement interpolée entre ces deux points. Ainsi, plus les discontinuités dans le relief sont importantes, plus la texture est étirée. Cette limitation rend difficile l'extension de notre méthode au rendu de bas-reliefs. Un autre problème du *relief mapping* est, comme le souligne [BD06], la présence d'artefacts dus à l'échantillonnage discret du rayon lors de la recherche d'intersections. Dans notre cas, le relief est vraiment faible par rapport à ce qui est généralement rendu à l'aide de cet algorithme, et ces artefacts sont donc limités. Cela nous permet d'ailleurs de réduire le nombre de pas pour la recherche linéaire afin d'accélérer encore un peu le rendu.

7. Conclusions et Perspectives

Dans ce papier, nous avons introduit une nouvelle chaîne de traitement pour numériser et restituer des peintures d'art. Cette méthode permet d'acquérir la géométrie et la couleur d'un tableau. Nous utilisons un rendu hybride basé sur deux algorithmes différents pour tenir correctement compte du relief de la peinture, selon son importance. Un mécanisme adaptatif est utilisé pour combiner les deux algorithmes. Le rendu temps réel est atteint, et pour un faible coût mémoire car la peinture n'est représentée que par un champ de hauteurs et trois textures pour son matériau bidirectionnel.

Actuellement, nous n'avons travaillé que sur un petit tableau. Notre chaîne pourrait être étendue pour permettre l'acquisition de grandes toiles, ne rentrant pas entièrement dans le champs du scanner, ce qui pose actuellement problème pour certains des pré-traitements. Enfin, étant donné que nous avons remarqué que le rendu par maillage était plus efficace pour les points de vue proches, nous voudrions ajouter un troisième niveau de rendu, utilisant le maillage lui même pour l'échelle la plus fine.

Références

[BD06] BABOUD L., DÉCORET X.: Rendering geometry with relief textures. In *Proc. of the 2006 conference on Graphics interface* (2006), pp. 195–201.

[Bli78] BLINN J. F.: Simulation of wrinkled surfaces. In *Proc. of SIGGRAPH '78* (1978), pp. 286–292.

[BM93] BECKER B. G., MAX N. L.: Smooth transitions between bump rendering algorithms. In *Proc. of SIGGRAPH '93* (1993), pp. 183–190.

[Coo84] COOK R. L.: Shade trees. In *Proc. of SIGGRAPH '84* (1984), pp. 223–231.

[DNGK97] DANA K. J., NAYAR S. K., GINNEKEN B. V., KOENDERINK J. J.: Reflectance and texture of real-world surfaces authors. In *Proc. of the Conference on Computer Vision and Pattern Recognition* (1997), p. 151.

[GAL03] GUIDI G., ATZENI C., LAZZARI S.: 3d optical scanning diagnostics for Leonardo Da Vinci's "Adorazione dei Magi" conservation. In *3DIM* (2003), pp. 110–115.

[GS03] GRATTONI P., SPERTINO M.: A mosaicing approach for the acquisition and representation of 3d painted surfaces for conservation and restoration purposes. *Mach. Vision Appl. 15*, 1 (2003), 1–10.

[GW07] GIEGL M., WIMMER M.: Unpopping: Solving the image-space blend problem for smooth discrete lod transitions. *Computer Graphics Forum 26* (2007), 46–49.

[HDKS00] HEIDRICH W., DAUBERT K., KAUTZ J., SEIDEL H.-P.: Illuminating micro geometry based on precomputed visibility. In *Proc. of SIGGRAPH '00* (2000), pp. 455–464.

[LD06] LARUE F., DISCHLER J.-M.: Automatic registration and calibration for efficient surface light field acquisition. In *Proc. of VAST06* (2006), pp. 171–178.

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proc. of SIGGRAPH '97* (1997), pp. 117–126.

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital Michelangelo project: 3d scanning of large statues. In *Proc. of SIGGRAPH '00* (2000), pp. 131–144.

[MLH02] MCALLISTER D. K., LASTRA A., HEIDRICH W.: Efficient rendering of spatial bi-directional reflectance distribution functions. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2002), pp. 79–88.

[MMK03] MESETH J., MÜLLER G., KLEIN R.: Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium* (2003), pp. 89–96.

[MMS*05] MULLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis, and rendering of bidirectional texture functions. *Computer Graphics Forum 24*, 1 (2005), 83–109.

[MWL*99] MARSCHNER S. R., WESTIN S. H., LAFORTUNE E. P., TORRANCE K. E., GREENBERG D. P.: Image-based BRDF measurement including human skin. In *Rendering Techniques '99, Proc. of the Eurographics Workshop* (1999), pp. 131–144.

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental analysis of BRDF models. In *Eurographics Symposium on Rendering* (2005), pp. 117–126.

[OBM00] OLIVEIRA M. M., BISHOP G., MCALLISTER D.: Relief texture mapping. In *Proc. of SIGGRAPH '00* (2000), pp. 359–368.

[POC05] POLICARPO F., OLIVEIRA M. M., COMBA J. L. D.: Real-time relief mapping on arbitrary polygonal surfaces. In *Proc. of the 2005 symposium on Interactive 3D graphics and games* (2005), pp. 155–162.

[RTG97] RUSHMEIER H. E., TAUBIN G., GUÉZIEC A.: Applying shape from lighting variation to bump map capture. In *Proc. of the Eurographics Workshop on Rendering Techniques '97* (1997), pp. 35–44.

[Tat06] TATARCHUK N.: Dynamic parallax occlusion mapping with approximate soft shadows. In *Proc. of the 2006 symposium on Interactive 3D graphics and games* (2006), pp. 63–69.

[TTK04] TOMINAGA S., TANAKA N., KOMADA T.: Imaging and rendering of oil paintings using a multi-band camera. In *Southwest Symposium on Image Analysis and Interpretation* (2004), pp. 6–10.

Figure 1: Gauche*: copie numérique du Tableau du Port dans une galerie virtuelle.* Droite*: illustration de notre mécanisme adaptatif. Pour les angles rasants, plus de boîtes sont sélectionnées pour éviter les erreurs de parallax dues au* bump mapping*.*

Effets de flou visuel pour la navigation en environnements virtuels en vue à la première personne

Sébastien Hillaire [1,2] et Anatole Lécuyer[2,3,4] et Rémi Cozot[1,2] et Géry Casiez[3,5]

Université de Rennes 1[1], IRISA[2], INRIA[3], Laboratoire de Physiologie de la Perception et de l'Action[4], Université de Lille 1[5]

Abstract

Cet article étudie l'utilisation d'effets visuels de flou lors d'une navigation en vue à la première personne dans des environnements virtuels. Premièrement, nous présentons un modèle de flou visuel dynamique basé sur deux types d'effet de flou: (1) un flou de profondeur qui simule la perception flou des objets hors focus, et (2) un flou de périphérie qui simule une perception flou des objets situés à la périphérie du champ de vision. Nous introduisons deux nouvelles techniques pour améliorer le flou de profondeur temps-réel: (1) un paradigme calculant automatiquement la distance focale, et (2) un filtrage temporel qui simule le phénomène d'accomodation. Deuxièmement, nous décrivons les résultats d'une expérience pilote menée afin étudier l'influence de tels effets de flou sur les performances et préférences de joueurs de jeux vidéo. De façon intéressante, il apparaît que les effets visuels de flou n'ont pas dégradé les performances des joueurs et qu'ils ont été préférés par la moitié des participants pour améliorer le côté ludique du jeu et le "gameplay". Nos résultats suggèrent donc que l'utilisation d'effets visuels de flou semble approprié pour les jeux vidéo et pour d'autres environnements virtuels avec une navigation en vue à la première personne.

This paper studies the use of visual blur effects for First-Person-Navigations in virtual environments. First, we introduce a model of dynamic visual blur based on two types of blur effect: (1) a Depth-of-Field blur which simulates the blurring of objects out of focus, and (2) a peripheral blur which simulates the blurring of objects located at the periphery of the field of vision. We introduce two new techniques to improve real-time DoF: (1) a paradigm to automatically compute the focal distance, and (2) a temporal filtering that simulates the accommodation phenomenon. Second, we describe the results of a pilot experiment conducted to study the influence of such blur effects on the performance and preference of video gamers. Interestingly, it seems that visual blur effects did not degrade performance of gamers and they were preferred by half of the participants to improve fun and game-play. Taken together, our results suggest that the use of visual blur effects could thus be suitable in videogames and in other virtual environments.

1. Introduction

La profondeur de champ de l'oeil humain est la plage des distances située près du point de focalisation, là ou les yeux perçoivent une image nette. Les objets situés devant ou derrière ce point de focalisation apparaissent flous. La profondeur de champ et son effet de flou associé sont des informations visuelles bien connues de la vision humaine [AS00] [MS02]. Dépourvues d'un effet de flou de profondeur, les images virtuelles peuvent parfois paraître trop synthétiques ou parfaites. Ainsi, le flou généré par le phénomène de profondeur de champ a très tôt été utilisé pour le rendu d'image par ordinateur [PC81]. Les effets de flou dus à la variation de la distance de focalisation sont aussi classiquement utilisés dans les films pour provoquer certaines sensations chez l'observateur ou attirer son attention.

Jusqu'ici, les effets visuels de flou n'ont pas encore été utilisés dans les applications de réalité virtuelle. Cependant, les capacités de programmation et la puissance de calcul des cartes graphiques actuelles rendent aujourd'hui possible le calcul de ces effets en temps réel. En conséquence, la prochaine génération de jeux vidéo va progressivement

intégrer ces effets dans leur moteur de rendu. Les implementations actuelles restent cependant limitées : le flou est seulement appliqué uniformément à l'arrière plan ou au premier plan de l'image. En outre, nous ne connaissons pas aujourd'hui l'influence de ces effets visuels de flou sur les performances et l'appreciation des utilisateurs. Ainsi, aujourd'hui, nous avons besoin de deux types d'investigation concernant l'utilisation d'effets visuels de flou en environnements virtuels : (1) concerant le développement de nouveaux modèles de flou en environnements virtuels tenant compte de l'interactivité et des contraintes temps réel, et (2) concernant l'évaluation de l'utilisation de ces effets visuels de flou en termes de performance et de préférence lors de navigations en environnement virtuel.

Dans la suite de cet article, nous concentrons notre étude sur des situations de navigation en environnements virtuels en vue à la première personne avec un champ visuel relativement restreint. Après avoir brièvement décrit un état de l'art sur les effets visuels de flou appliqués au images générées par ordinateur, nous introduirons un nouveau modèle de flou visuel dynamique adapté à la navigation temps réel en environnements virtuels. Ce modèle de flou est basé sur l'effet de profondeur de champ avec un calcul automatique de la distance de focalisation combiné à une adaptation temporelle. Ce modèle inclut également un flou périphérique qui applique un flou sur au contour du champ de vision pour inciter l'utilisateur à regarder au centre de l'écran. Ensuite, nous exposerons une expérience menée pour étudier l'influence de ce modèle de flou sur les performances de joueurs durant des sessions multi-joueurs d'un jeu de tir en vue à la première personne (First-Person-Shooter : FPS). Dans ce but, notre modèle de flou a été implémenté dans le moteur de rendu d'un célèbre jeux video [IDS07]. Le papier se termine par une conclusion générale et une description des travaux futurs.

2. État de l'art

La simulation de flou visuel a été introduite en informatique graphique afin d'améliorer l'aspect photo-réaliste des images synthétiques. Potmesil et Chakravarty [PC81] ont été les premiers à proposer une simulation de lentille optique dans le but de simuler le flou dû à l'effet de profondeur de champ. Leur algorithme utilise l'image nette originale et la profondeur de chaque pixel. Ensuite, un algorithme post-rendu est utilisé pour appliquer un effet de flou. La simulation de la lentille permet de calculer le niveau de flou de chaque pixel selon sa profondeur. En simulant une lentille, un point situé hors focus devient un cercle lorsqu'il est projeté au travers de la lentille sur le plan de projection. Le cercle résultant est appelé le *cercle de confusion*. Le diamètre de ce cercle de confusion correspond au niveau de flou [PC81]. Après ce travail pionnier, la plupart des chercheurs ont utilisé ce modèle de lentille pour calculer le niveau de flou des pixels d'une image dû à la profondeur de champ [Dem04]. Précisemment, Barsky [Bar04] a introduit le concept de rendu réaliste de la vision ('vision-realistic rendering') qui réplique toutes les caractéristiques du système visuel et optique d'un individu. Barsky peut précisement simuler l'image fovéale en acquérant les propriétés du système visuel de sujets humains à l'aide d'un abberomètre.

Le principal problème des algorithmes de flou de profondeur est un artefact visuel appelé *débordement de couleur* visible le long des discontinuités de profondeur. Cet artefact est dû à l'inclusion de la couleur des contours des objets nets lors du calcul du flou des objets situés hors focus. Les algorithmes de flou de profondeur existants se différencient par la façon dont ils calculent le flou lui-même et évitent ce problème. Le lecteur intéréssé pourra consulter sur ce sujet l'état de l'art sur les techniques de flou de profondeur publié récemment par Demers [Dem04]. Actuellement, les différentes techniques peuvent être divisées en trois catégories : les techniques de dispersion [KZB03], de regroupement [Dem04] et de diffusion [KLO06]. Par exemple, la technique de regroupement (aussi appelée technique d'application inverse) utilise les pixels de l'image finale. Pour chaque pixel de l'image, l'algorithme rassemble et mélange la couleur des pixels situés à l'intérieur de son cercle de confusion. Une simple comparaison des profondeurs des pixels lors du rassemblement permet d'éviter le phénomène de débordement de couleur. Cette approche peut être facilement implémentée sur carte graphique [Dem04].

D'autres effets visuels de flou peuvent être utilisés pour améliorer l'apparence réaliste des images générées par ordinateur. Par exemple, le flou de périphérie [Ans98] simule la perte d'acuité visuelle à la périphérie du champ de vision [SDR*95]. Le flou de mouvement [ML85] simule l'image perçue lorsque des objets de la scène se déplacent rapidement. Il correspond à l'intégration des images des positions successives d'un objet en mouvement se déplaçant devant une caméra lorsque l'obturateur reste ouvert. Des implémentations récentes ont démontré qu'il était également possible d'utiliser de tels effets en temps réel sur carte graphique [WZ96].

À la connaissance des auteurs, l'utilisation d'effets visuels de flou en temps réel dans des applications de réalité virtuelle a seulement été suggérée par Rokita [Rok96], mais n'a pas été implémentés. Rokita propose une technique de rassemblement couplée à un modèle de lentille pour simuler un effet de flou de profondeur. Il suggère que de tels effets de flou sont importants pour les applications de réalité virtuelle. Rokita suggère également d'utiliser un système de suivi du regard afin de calculer le point de focalisation de l'utilisateur à l'écran, mais aucun détail d'implémentation supplémentaire n'est cependant fournis.

3. Effets visuels de flou pour la navigation en environnements virtuels en vue à la première personne

Dans cette partie, nous décrivons le modèle dynamique de flou visuel que nous proposons pour la navigation en environnements virtuels avec une vue à la première personne. Ce modèle est basé sur deux effets de flou : (1) un effet de flou de profondeur et (2) un effet de flou de périphérie.

3.1. Effet de flou de profondeur

L'effet de flou de profondeur consiste à appliquer effet de flou aux pixels des objets situés devant ou derrière le point de focalisation de l'utilisateur. Ce point de focalisation est associé à la distance de focalisation f_d, c'est-à-dire, la distance entre les yeux et le point de focalisation. Dans des applications interactives et temps-réel, la distance de focalisation doit être calculée dynamiquement. En effet, l'utilisateur peut être amené à regarder différents objets situés à des distances différentes. Ainsi, une telle application requiert : (1) le calcul automatique du point de focalisation et (2) la simulation du phénomène d'accomodation du système visuel humain.

3.1.1. Utilisation d'un modèle de lentille

Nous utilisons le modèle classique de lentille pour calculer le niveau de flou de chaque pixel dû au phénomène de profondeur de champ [PC81]. Dans ce modèle, le niveau de flou, c'est-à-dire, le diamètre du cercle de confusion $DCoC_d$ est donné par :

$$DCoC_d = \left| \frac{D \times f \times (f_d - z)}{f_d \times (z - f)} \right| \quad (1)$$

Où D est le diamètre de la lentille, f la distance focale de la lentille, f_d la distance de focalisation et z la profondeur du point.

3.1.2. Calcul automatique de la distance de focalisation

Une manière optimale de déterminer en temps réel la distance de focalisation serait d'utiliser un système de suivi du regard. Cependant, de tels équipements sont aujourd'hui encore chers, complexes et non accessibles au grand public. Ainsi, en l'absence de tels systèmes, nous proposons d'utiliser un paradigme utilisé dans les jeux de type FPS. Dans les FPS, l'utilisateur manipule, avec la souris et le clavier, un viseur virtuel toujours situé au centre de l'écran. Dans ce cas, nous supposons que l'utilisateur regarde principalement la partie de l'écran proche du viseur. En utilisant un système de suivi du regard, Kenny et al. [KKD*05] ont ainsi remarqué que les joueurs de FPS regardaient plus de 82% du temps la zone centrale de l'écran. Ainsi, nous introduisons la notion de zone d'auto-focus, c'est-à-dire, une zone positionnée au centre de l'écran que l'utilisateur est supposé préférer regarder. Cette zone de focus rappelle les systèmes d'auto-focus utilisés par les caméras dont le but est de fournir une distance de focalisation appropriée lors de la prise de vue. Les systèmes de Fujifilm (par exemple, FinePix S6000fd et FinePix F31fd) utilisent une approche sémantique afin que les visages humains détectés dans l'image apparaissent nets [Fuj07].

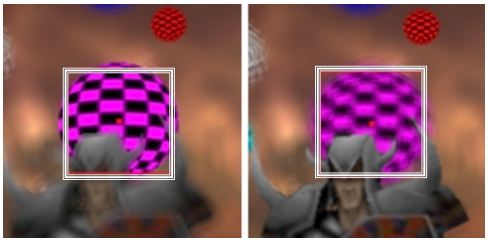


Figure 1: *Flou de profondeur utilisant une zone de auto-focus sans (gauche) et avec (droite) pondération sémantique.*

La profondeur de chaque pixel à l'intérieur de la zone d'auto-focus est tout d'abord calculée dans un tampon auxiliaire. Dans le cas d'un jeu vidéo de type FPS, certains objets de l'environnement ont plus d'importance que d'autres (ennemis, objets bonus). Pour tenir compte de cela, nous proposons une pondération sémantique de la profondeur de chaque pixel. Le poid sémantique est utilisé pour accroître le poids des pixels correspondants à des objets importants. Ainsi, nous ajoutons aux objets virtuels une propriété correspondant à sa valeur sémantique. La valeur sémantique peut varier entre $WSmin$ et $WSmax$. La figure 1 illustre l'utilisation de la pondération sémantique. Dans cet exemple, la valeur sémantique du personnage au premier plan est plus grande que celle de la sphère décorative en second plan. Au final, même si le personnage couvre moins de pixels que la sphère dans la zone d'auto-focus (moins d'un quart), le focus est tout de même effectué sur lui.

De plus, nous modifions légèrement le poids des profondeurs suivant la distance des pixels par rapport au centre de l'écran. Cela peut être réalisé en utilisant une fonction gaussienne qui donne une valeur $WGmax$ au centre de la zone d'auto-focus et $WGmin$ sur les bords. Finalement, la distance de focalisation est calculée en utilisant l'équation 2 avec $WS(p)$ le poids sémantique du pixel p, $WG(d)$ le poids gaussien spatial pour une distance d et $d2AC(p)$ la distance du pixel p au centre de la zone d'auto-focus. Dans notre implémentation finale, la longueur de l'arête de la zone d'auto-focus était de 33 pixels (c'est-à-dire, 11.9 mm pour un écran 17' avec une résolution de 1024, $WGmin = 0.7$, $WGmax = 1.0$, $WSmin = 0.004$ et $WSmax = 1$.

$$f_d = \frac{\sum_{p \in area} WS(p) \times WG(d2AC(p)) \times depth(p)}{\sum_{p \in area} WS(p) \times WG(d2AC(p))} \quad (2)$$

3.1.3. Simulation du phénomène d'accomodation

Dans le système visuel humain, le passage d'un point de focalisation à un autre nécessite un temps d'adaptation. Nous

proposons donc de simuler ce phénomène d'accomodation pour notre modèle de flou de profondeur. Pour cela, nous ajoutons un filtre temporel lors du calcul de la distance de focalisation. Après des tests préliminaires, nous avons choisi un filtre passe-bas donné par l'équation 3 , avec $\tau = (\pi \times fc)/2$, f_c étant la fréquence de coupure, T_e la période d'échantillonage, $\overline{f_d}(n)$ la distance de focalisation filtrée de l'image n et $f_d(n)$ la distance de focalisation calculée par le système d'auto-focus avant filtrage. Dans notre implémentation, $T_e = 1/70$ seconde et $f_c = 5Hz$.

$$\overline{f_d}(n) = \left(f_d(n) + \frac{\tau}{T_e} \overline{f_d}(n-1) \right) \frac{1}{1+\frac{\tau}{T_e}} \qquad (3)$$

3.2. Effet de flou de périphérie

Le flou de périphérie simule le fait que la netteté des objets perçus décroit lorsqu'ils s'éloignent du centre de l'image perçue. Cet effet est indépendant du flou de profondeur. Le flou de périphérie est ajouté comme un effet supplémentaire qui permet d'augmenter les sensations des utilisateurs. Cependant, l'objectif principal du flou de périphérie est d'inciter l'utilisateur à regarder au centre de l'écran, c'est-à-dire, à l'intérieur de la zone d'auto-focus.

Figure 2: *Utilisation du flou de périphérie.*

Le calcul du niveau de flou de périphérie $DCoC_p$ est donné par l'équation 4 dans laquelle $\mathbf{z}$ est la direction du regard (dans notre cas, la direction de la caméra) et $\mathbf{p}$ la direction normalisée ayant pour origine le centre de la caméra et pour extrémité la surface affichée par le pixel dans le repère de la caméra. Dans notre implémentation finale, $n = 2$. La figure 2 présente un exemple d'image obtenue avec notre effet de flou de périphérie.

$$DCoC_p = \left(\sqrt{\frac{1}{\mathbf{z} \cdot \mathbf{p}} - 1} \right)^n \qquad (4)$$

3.3. Calcul du flou visuel final

Une fois que les calculs du flou de périphérie et de profondeur sont terminés, nous pouvons calculer le niveau total de flou pour chaque pixel, c'est-à-dire, le diamètre final du cercle de confusion $DCoC_f$ en utilisant l'équation 5 avec D_{max} le niveau maximum de flou (le diamètre maximum du cercle de confusion de chaque pixel). Dans notre implémentation, $D_{max} = 12$.

$$DCoC_f = D_{max} \times \min(1, DCoC_d + DCoC_p) \qquad (5)$$

Le flou peut finalement être appliqué à l'image en mélangeant la couleur de chaque pixel avec la couleur des pixels situés à l'intérieur de son cercle de confusion. Lire la couleur de chaque pixel situé à l'intérieur d'un cercle de confusion résulterait en un taux de rafraîchissement du rendu à l'écran trop lent. Afin de maintenir un taux de rafraîchissement élevé, les pixels sont sélectionnés aléatoirement à l'intérieur du cercle de confusion suivant une loi de distribution poisson-disque comme dans [Dem04]. Finalement, nous effectuons une comparaison des profondeurs pour chaque pixel afin d'éliminer le problème de débordement de couleur.

3.4. Implémentation

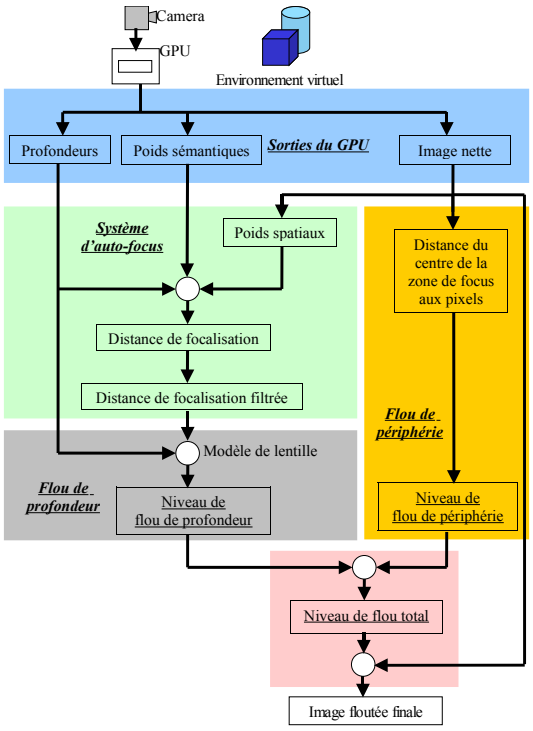


Figure 3: *Architecture logicielle et calcul des effets de flou visuel.*

L'architecture finale de notre modèle de flou est représentée sur la figure 3. Nous calculons sur GPU les trois composantes par pixel nécessaires à notre modèle de flou : le poid sémantique (image A de la figure 4), la profondeur (image B) et l'image nette de l'environnement virtuel. Ensuite, le système d'auto-focus utilise ces trois composantes pour déterminer la distance de focalisation (voir section 3.1.2). La distance de focalisation est alors filtrée par un filtre passe-bas (phénomène d'accomodation). Ensuite, l'algorithme de flou de profondeur calcule le niveau de flou par pixel, du à l'effet de profondeur de champ, en utilisant leur profondeur et la distance de focalisation filtrée en entrée du modèle de lentille (équation 1). Parallèlement, le flou de périphérie est lui aussi calculé en utilisant l'équation 4. Le niveau total de flou par pixel (image C) est ensuite calculé suivant l'équation 5 en utilisant les niveaux de flou de profondeur et de périphérie de chaque pixel. Finalement, l'algorithme de flou est appliqué à l'image nette, en utilisant le niveau total de flou par pixel, afin d'obtenir l'image finale de l'environnement virtuel (image D).

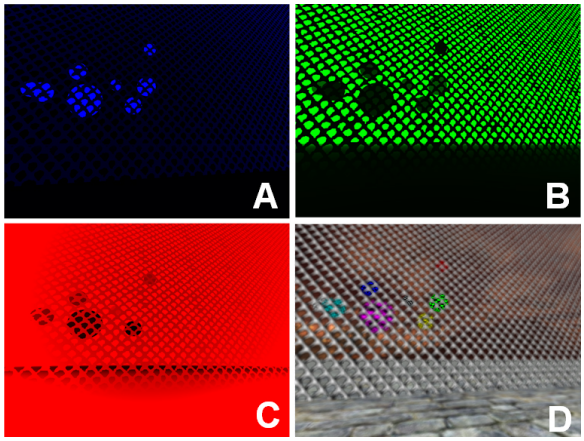


Figure 4: *Illustration du processus de calcul du flou. (A) poids sémantiques, (B) profondeurs, (C) niveau de flou total et (D) image finale. Dans ce cas, l'utilisateur est situé devant un grillage; il peut tout de même focaliser sur les objets importants (sphères) situés derrière le grillage grâce à l'utilisation de la pondération sémantique.*

Notre effet de flou a été implémenté dans le moteur de rendu temps réel du célèbre jeu video *Quake III Arena* [IDS07]. Nous avons utilisé un Dell Master Precision G1 J502-03, Intel PentiumD CPU à 3.4Ghz avec 2Go de RAM et une carte graphique nVidia Quadro FX 3450/4000 SDI. Les performances étaient de 260 images par seconde sans notre modèle de flou activé. Elles étaient au minimum de 70 images par seconde lorsque nous utilisions notre modèle de flou sur un rendu effectué à une résolution de 1024×768 pixels. La photo d'écran de la figure 5 montre le résultat obtenu lors du rendu d'une scène avec le flou de profondeur et le flou de périphérie activés.

3.5. Conclusion préliminaire

Notre modèle visuel de flou intéractif combine deux types d'effet de flou : (1) un flou de profondeur et (2) un flou de périphérie. Il a été développé dans le cadre d'une navigation temps réel en environnement virtuel avec une vue à la première personne. Pour rendre la technique de flou de profondeur plus adaptée à ce type de navigation dans des applications de réalité virtuelle, nous avons introduit deux techniques : (1) un calcul automatique de la distance de focalisation et (2) un filtrage temporel de cette même distance afin de simuler le phénomène d'accomodation du système visuel humain.

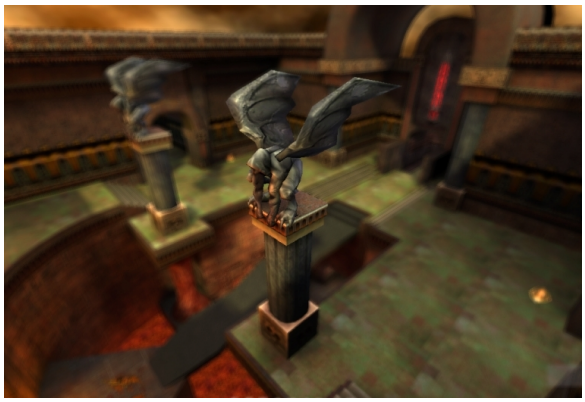


Figure 5: *Le jeu vidéo Quake III avec notre modèle de flou dynamique implémenté.*

Ces effets ont été implémentés avec succès dans le moteur de rendu open-source d'un jeu vidéo [IDS07]. Ce jeu vidéo et la configuration mentionée précedement ont été directement utilisés pour évaluer l'influence de nos effets visuels de flou sur les performances de joueurs. La section suivante décrit le protocole et les résultats de cette expérience pilote.

4. Experience pilote

Une expérience pilote a été menée afin d'étudier l'influence de ces effets visuels de flou sur les performances et les préférences subjectives de joueurs de jeu vidéo. Les participants à cette expérience devaient participer à des sessions multi-joueurs de combat du jeu vidéo Quake III Arena. Les effets de flou étaient activés pendant la moitié des sessions et les performances des joueurs étaient enregistrées dans les deux cas, c'est-à-dire, avec ou sans flou. Un questionnaire subjectif était fourni aux participants pour connaître leur préference concernant le coté ludique du jeu ("fun"). Avant cette expérience, nous pouvons formuler les hypothèses suivantes :

1. H1 : le flou visuel dégrade les performances des participants (puisque l'image est en partie flou).

2. H2 : le flou visuel peut améliorer le coté "fun" du jeu (puisque c'est un effet visuel supplémentaire).

4.1. Dispositif

Nous avons utilisé notre version du moteur de Quake III modifiée afin de supporter notre modèle de flou. Les pondérations sémantiques et spatiales du système d'auto-focus étaient toutes les deux activées. Le poids sémantique des ennemis était positionné à $WSmax$ et toutes les autres valeurs sémantiques à $WSmin$. Les autres valeurs numériques que nous avons utilisées sont données dans les sections précédentes. La carte de Quake III que nous avons utilisée est nommée "Q3DM7".

Nous avons utilisé 8 PC identiques en réseau local avec la même carte graphique connectée au même écran (figure 6) et configurée à une résolution identique (détail dans la section 3.4). Aucun ralentissement n'était perceptible. Des casques audio et souris infra-rouge étaient fournis pour chaque participant.

Figure 6: *Photo de la salle d'expérience.*

4.2. Tâche et stimuli

La tâche constistait à jouer à Quake III, un FPS, en mode "death-match" (chaque joueur se battait contre tous les autres). Il était demandé aux participants d'être aussi précis que possible lors de leur tir tout en utilisant le moins de munitions possible. Pour réduire la variabilité entre les joueurs, ceux-ci ne possédaient qu'une seule arme (une mitrailleuse) avec des munitions illimitées. Tous les bonus de vie et les bonus spéciaux avaient été retirés de la carte. Le niveau de vie de départ était élevé à 200, au lieu de 100, afin d'allonger la durée des combats.

4.3. Participants

Trente deux sujets (29 hommes, 3 femmes) avec une moyenne d'âge de 25.3 (SD=5.5) ont participé à cette expérience. Sur les 32 sujets, 14 n'avaient jamais eu d'expérience antérieure avec un jeu de type FPS tandis que les 18 autres participants avaient déjà une expérience modérée des jeux de type FPS, voir très important.

4.4. Conception

Un plan expérimental intra-sujet a été utilisé. La variable indépendante était l'effet visuel (effet de flou activé ou désactivé). L'expérience durait 120 minutes en incluant les pauses. Les participants étaient divisés en quatre groupes de 8 sujets chacun. L'expérience était ensuite divisée en deux parties.

La première partie était une session d'entraînement avec ou sans flou suivie de la vrai expérience avec une première session de 15 minutes dans les mêmes conditions que la session d'entraînement, puis d'une seconde session de 15 minutes avec la condition inverse. Après ces deux sessions, les participants devaient remplir un questionnaire subjectif.

La seconde partie était un test de performance incluant 6 sessions. Pour chacune des 6 sessions, 4 sujets jouaient avec l'effet de flou activé et les 4 autres jouaient sans effet de flou. La condition de flou était automatiquement changée pour chaque joueur lorsqu'une nouvelle session était lancée. Dans chaque partie de l'expérience, l'ordre de présentation était changé pour chaque participant de chaque groupe.

4.5. Résultats

4.5.1. Performance

Les variables dépendantes étaient enregistrées pour chaque participant à la fin de chacune des 6 sessions de test. Ces variables étaient : le nombre d'ennemis que le joueur a tué durant la session (*Frags*), le nombre de fois où il est mort (*Mort*), le nombre total de balles tirées (*Tirs*) et La précision (*Precision*, en pourcentage) qui représente le rapport entre le nombre de tirs réussis (ayant atteint un ennemi) sur le nombre total de tires effectués.

De façon intéressante, la performance des joueurs lorsqu'ils étaient exposés aux effets de flou semble similaire à leur performance lorsqu'aucun effet de flou n'est activé (voir Table 1). En effet, la moyenne et la déviation standard de chacune des variables dépendantes sont très proches.

	Avec flou		Sans flou	
	Moyenne	DS	Moyenne	DS
Frags	28.7	9.9	28.5	10.0
Mort	28.6	4.7	28.7	4.6
Tir total	2694	496	2653	509
Précision (%)	29.2	5.3	29.1	5.5

Table 1: *Moyennes et déviations standards (DS) pour chaque variable dépendante du test de performance, avec et sans effet de flou.*

Une analyse à mesures répétées de variance (ANOVA) a été effectuée sur les données. L'ANOVA a d'abord montré que l'ordre de présentation des effets visuels de flou et les groupes n'avaient pas d'effet significatif sur les différentes variables dépendantes. Ensuite, aucun effet significatif de

l'effet visuel de flou sur les variables dépendantes n'a été trouvé. Une analyse statistique de puissance a ensuite été menée [Coh89]. Elle a montré que notre analyse a une puissance supérieure à 80% pour détecter un effet important (taille de l'effet=0.4). Ceci est suffisant pour conclure qu'il n'y a pas d'effet important du flou sur les performances des sujets.

Ainsi, nos résultats tendent à montrer que les participants peuvent maintenir un niveau similaire de performance lorsque notre modèle de flou est activé comparé à la situation où notre modèle de flou est désactivé.

4.5.2. Questionnaire et retours d'utilisateurs

Nous avons pu rassembler de nombreuses informations utiles concernant l'appréciation des effets de flou durant le jeu à partir des questionnaires subjectifs des 32 participants.

Tout d'abord, après les deux sessions de 15 minutes, une large majorité des participants (84%, c'est-à-dire, 27 participants) ont été capables de discerner une différence entre les sessions. Ensuite, 78% (25/32) ont été capables de formuler explicitement qu'un effet de flou était appliqué à l'image. Quelques participants n'avaient pas clairement compri la signification et/ou le mode de calcul de cet effet de flou : "effet de flou bizarre", "flou non systématique", "flou intermittent".

Le questionnaire final d'appréciation globale n'a pas montré une tendance significative concernant une potentielle appréciation de l'effet de flou. En effet, les participants étaient partagés concernant l'amélioration du *réalisme* de la scène virtuelle (14 préféraient avec le flou, 16 préféraient sans et 2 n'avaient aucune préférence), l'amélioration du *fun* (13 préféraient avec, 13 sans, 6 pas de préference), l'amélioration de la *perception des profondeurs* (16 préféraient avec, 13 sans, 3 pas de préference) et l'amélioration du *sentiment de présence* (11 préféraient avec, 11 sans, 10 pas de préference).

Les personnes qui ont préféré le jeu lorsque l'effet de flou était activé pouvaient être très enthousiastes : "c'était plus intense avec le flou", "on en devient dépendant", "c'est difficile de retourner à la situation sans flou", "c'est plus fun", "plus réaliste", "plus joli", "le gameplay est amélioré", "le jeu est plus crédible". Ainsi, il semble donc que de nombreux participants soient prêt à sélectioner et activer de flou pour une utilisation future du jeu vidéo.

Les personnes qui ont préféré le jeu sans l'effet de flou se sentaient généralement fatiguées (principalement au niveau des yeux et de la tête). Elles estimaient que le flou était responsable de leur fatique (31%, 10/32). Certaines d'entre elles, spécialement les joueurs experts, ont trouvé que le flou était un "incomfort". Elles l'ont perçu comme "trop fort", "surtout sur les bords". De plus, ces personnes étaient gênées par le flou lorsqu'elles exploraient l'image à la recherche de cibles/ennemis à l'écran.

D'autre part, une majorité des participants a déclaré que la présence du flou n'a pas modifié leur stratégie lorsqu'ils combattaient dans le jeu. Parmis les 5 participants restants, les commentaires étaient paradoxaux. Certains d'entre eux déclaraient "se rapprocher des adversaires" avec le flou activé alors que les autres le trouvaient plus adapté à une "stratégie de tireur embusqué" ("sniper").

4.6. Discusssion

Nos résultats montrent, d'une part, que l'activation des effets de flou ne semblent pas abaisser les performances des participants durant le jeu et, d'autre part, que la moitié des participants ont préféré les effets de flou pour améliorer le réalisme ou le fun.

Les premiers résultats (concernant les performances constantes des participants dans les deux conditions) contredisent heureusement notre première hypothèse H1. Cela semble être cohérent avec le fait que les participants ont exprimé que leur stratégie était similaire, que les effets de flou soient activés ou pas. Comme les participants ne modifiaient pas leur stratégie, ils pouvaient maintenir le même niveau de performance au cours de chaque session.

Notre seconde hypothèse était que les effets de flou pouvaient améliorer l'appréciation des sujets concernant certains critères de jeu tel que le fun. Cela est confirmé par le fait que la moitié des participants ont préféré la présence d'effets de flou en termes de fun, de présence et de réalisme de l'environnement virtuel. Cela semble suffisant pour que ces effets de flou soient recommandés, par exemple en tant qu'option dans les jeux vidéo. Mais nous avons aussi remarqué que le même nombre de participants n'avait pas apprécié les effets au cours des sessions, bien que leur performance ne soit pas dégradée. Les commentaires de plusieurs participants suggèrent qu'ils ne regardaient pas constamment au centre de l'écran, c'est-à-dire, à l'intérieur de la zone d'autofocus. Il y a au moins deux phases qui peuvent être identifiées. Lors de la première phase, le joueur tire sur les ennemis avec son viseur et regarde donc bien au centre de l'écran. Dans ce cas, la distance de focalisation calculée est adaptée au point de focalisation de l'utilisateur. Dans une seconde phase, le joueur explore l'image pour trouver et identifier des ennemis. Pendant cette phase, la distance de focalisation calculée avec notre modèle peut ne pas correspondre au point de focalisation de l'utilisateur à l'écran. Pour certains utilisateurs, cette situation était problématique et générait un inconfort et une fatigue. Cependant, cette fatigue visuelle ne les empêchait pas de maintenir leur niveau de performance constant. Pour les autres utilisateurs, cette situation n'était apparement pas un problème et le jeu pouvait tout de même gagner en "fun" et en présence.

Au cours de notre évaluation expérimentale, nous avions volontairement accentué le flou pour que les participants le perçoivent mieux. Cependant, plusieurs participants se sont

plains que le flou était parfois trop fort. Ceci peut expliquer la fatigue observée et exprimée par certains d'entre eux. Cela suggère aussi que l'intensité du flou doit être réglée avec prudence et que les utilisateurs doivent avoir la possibilité de régler le niveau du flou dans leur application.

5. Conclusion

Nous avons étudié l'utilisation d'effets visuels de flou pour la navigation en vue à la première personne en environnement virtuel. Un modèle dynamique de flou visuel a tout d'abord été introduit. Ce modèle est basé sur deux types de flou : un flou de profondeur qui simule le flou d'une image dû au phénomène de focalisation du système visuel humain et un flou de périphérie qui simule le flou naturellement présent à la périphérie de l'image perçue par un humain. Pour rendre ce modèle de flou plus approprié et utilisable dans des applications de réalité virtuelle, nous avons développé une technique de calcul automatique de la distance de focalisation combinée à un filtrage temporel de cette distance de focalisation pour simuler le phénomène d'accomodation.

Une expérience pilote a été conduite et a montré que l'activation de notre modèle de flou visuel ne semble pas déteriorer les performances des joueurs au cours de plusieurs sessions multi-joueur d'un jeu de type FPS. De plus, la moitié des participants ont préféré jouer quand l'effet de flou était activé. Les effets de flou étaient considérés comme une amélioration du réalisme du retour visuel et du fun. Ainsi, nos résultats suggèrent que l'utilisation d'effets visuels de flou peut être pertinente pour des jeux vidéo et d'autres applications de réalité virtuelle.

Travaux futurs. Les techniques décrites dans ce papier sont compatibles avec l'utilisation d'un système de suivi du regard. Ainsi, comme suggéré par Rokita [Rok96], des travaux futurs seraient nécessaires pour évaluer notre modèle de flou calculé avec un tel système. De plus, quelques participants ont suggéré d'utiliser cette technique dans d'autres applications et/ou conditions. D'autres expériences devraient ainsi être menées pour évaluer l'utilisation de nos techniques dans d'autres environnements virtuels, avec d'autres tâches et scénarios. Enfin, il serait intéressant d'évaluer l'influence de paramètres tels que le champ de vision et le dispositif d'interaction utilisé sur nos résultats.

Remerciements. Les auteurs tiennent à remercier toutes les personnes ayant participé à l'expérience pilote pour leur gentillesse et leur patience.

References

[Ans98] ANSTIS S.: Picturing peripheral acuity. *Perception 27* (1998), 917–825.

[AS00] ATCHISON D., SMITH G.: *Optics of the Humans eye*. Elsevier Health Science, 2000.

[Bar04] BARSKY B. A.: Vision-realistic rendering: simulation of the scanned foveal image from wavefront data of human subjects. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization* (New York, NY, USA, 2004), ACM Press, pp. 73–81.

[Coh89] COHEN J.: *Statistical Power Analysis for the Behavioral Sciences*, second ed. Elsevier Health Science, 1989.

[Dem04] DEMERS J.: Depth of field : A survey of techniques. 375–390.

[Fuj07] FUJIFILM:, 2007. http://www.facedetection.fujifilmusa.com/.

[IDS07] IDSOFTWARE: Quake III Arena, 2007. http://www.idsoftware.com/.

[KKD*05] KENNY A., KOESLING H., DELANEY D., MCLOONE S., WARD T.: A preliminary investigation into eye gaze data in a first person shooter game. In *Proc. 19th European Conference on Modelling and Simulation* (Riga, Latvia, 2005), Y. Merkuryev R. Z. . E. K., (Ed.), ECMS, pp. 733–740.

[KLO06] KASS M., LEFOHN A., OWENS J. D.: *Interactive Depth of Field Using Simulated Diffusion*. Tech. Rep. 06-01, Pixar Animation Studios, Jan. 2006.

[KZB03] KRIVANEK J., ZARA J., BOUATOUCH K.: Fast depth of field rendering with surface splatting. In *Proceedings of Computer Graphics International 2003* (2003).

[ML85] MAX N. L., LERNER D. M.: A two-and-a-half-d motion-blur algorithm. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 85–93.

[MS02] MATHER G., SMITH D.: Blur discrimination and its relation to blur-mediated depth perception. *Perception 31*, 10 (2002), 1211–9.

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM Press, pp. 297–305.

[Rok96] ROKITA P.: Generating depth-of-field effects in virtual reality applications. *IEEE Computer Graphics and Applications 16*, 2 (1996), 18–21.

[SDR*95] SERENO M., DALE A., REPPAS J., KWONG K., BELLIVEAU J., BRADY T., ROSEN B., TOOTELL R.: Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging. *Science 268*, 5212 (May 1995), 889–893.

[WZ96] WLOKA M. M., ZELEZNIK R. C.: Interactive real-time motion blur. *The Visual Computer 12*, 6 (1996), 283–295.

Navigation 3D virtuelle : positionnement et technique de sélection dans un affichage volumétrique complexe

Thomas Jund, David Cazier et Dominique Bechmann

LSIIT, UMR CNRS 7005, Université Louis Pasteur, Strasbourg, France

Résumé

Dans le cadre du développement de simulateurs chirurgicaux en environnement immersif, de nouveaux besoins se font sentir en ce qui concerne les outils d'interaction et de visualisation. Nous proposons dans cet article cinq outils d'interaction neufs ou adaptés à notre environnement de navigation virtuelle : un WIM à 6 degrés de liberté, le fil d'Ariane, une ligne de visée adaptative, un plan de coupe associé au WIM et une nouvelle métaphore, celle de la chandelle. Nous montrons comment ces outils s'articulent entre eux, pour améliorer la navigation dans des réseaux vasculaires denses.

1. Introduction

La réalité virtuelle (RV) est utilisée dans de nombreux domaines, depuis la visualisation de scènes simples, jusqu'à la navigation dans des environnements virtuels complexes, en passant par les simulateurs qui demandent un grand réalisme des scènes et donc une grande richesse des modèles utilisés. Le contexte de notre travail est le développement d'une application de simulation d'angioscopie, c'est-à-dire d'exploration de la paroi interne de vaisseaux sanguins. Notre support de réalité virtuelle est un Workbench utilisant la vision stéréoscopique et des outils d'interaction à six degrés de liberté.

Plus précisément il s'agit de familiariser des médecins aux déplacements d'une caméra – ou de tout autre outil – dans le système vasculaire d'un patient. Pour le moment, le simulateur propose deux modes de navigation : la navigation automatique qui permet de réaliser une *visite virtuelle* sur un chemin sélectionné et la navigation interactive qui permet de déplacer la caméra, en simulant la manipulation des instruments d'angioscopie et en détectant les collisions éventuelles avec les parois. Nous ne mettons pas en oeuvre de système de retour de force ni d'interfaces s'approchant du matériel que manipule un médecin dans cette simulation.

Dans le domaine de la RV, lorsqu'on développe une application, il est courant de classer les méthodes d'interaction proposées, en quatre grandes classes [Bow99] : les méthodes de navigation, de désignation, de manipulation et de contrôle d'application. Nous nous intéressons ici particulièrement aux problématiques liées à la navigation et à la désignation, celles-ci se révélant, en effet, cruciales dans le contexte de l'angioscopie.

Au niveau de l'interaction, la première difficulté provient du besoin de pouvoir positionner précisement des points à l'intérieur des vaisseaux. Dans le cas de la navigation libre, il s'agit de placer la caméra près de la zone d'intérêt. Dans celui de la visite automatique, il s'agit de définir les points de passage d'un trajet. Or l'environnement dans lequel est plongé l'utilisateur est particulièrement complexe et dense. Les figures 1 et 8 montrent des exemples de réseaux vasculaires dans lesquels on souhaite naviguer. Si la vue d'ensemble – ou vue extérieure – est relativement *lisible*, notre application nécessite de choisir précisément des zones à l'intérieur de ces vaisseaux. Par exemple, on peut avoir besoin de sélectionner une petite veinule à explorer ou une zone spécifique près de la paroi interne d'une grosse artère, pour visualiser une lésion qui s'y trouve.

Dès lors différentes difficultés apparaissent. Premièrement, les occlusions sont nombreuses. Beaucoup de veinules ou de petites artères sont cachées par des vaisseaux plus gros et restent invisibles même si l'on change de point de vue en tournant autour du système vasculaire. Deuxièmement, la densité des vaisseaux et leur enchevêtrement – mal rendu sur une image fixe – fait qu'il est difficile, sans aide visuelle, de les positionner les uns par rapport aux autres en profondeur

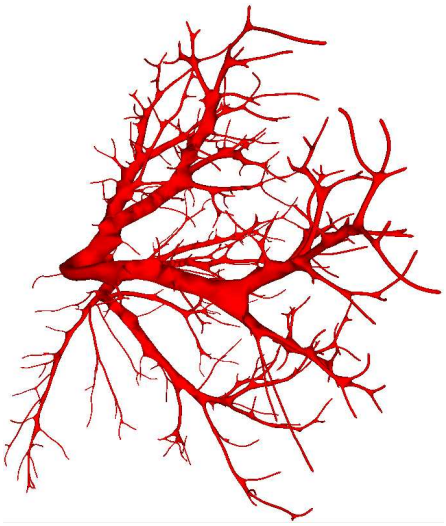


Figure 1: *Un système vasculaire complet.*

(c'est-à-dire de savoir quel vaisseau passe devant ou derrière tel autre). Cela pose de grandes difficultés lorsqu'il s'agit de réaliser une sélection précise dans des zones se situant à différentes profondeurs.

Le second type de difficultés soulevées par notre simulateur est lié à la possibilité donnée à l'utilisateur de se repérer ou de se positionner dans la simulation. En effet, une fois *plongés* à l'intérieur des vaisseaux, il est très difficile de s'orienter correctement. La figure 2 montre bien cette impression de labyrinthe donnée par la vue interne, même si l'affichage sur un dispositif de RV permet d'avoir une meilleure perception de la profondeur. Dans ce type d'immersion, il n'existe pas de repère visuel permettant d'identifier son propre positionnement, les notions de directions classiques (haut/bas, droite/gauche, devant/derrière) sont inadaptées, la caméra pouvant tourner sur elle-même. Par ailleurs, il n'existe par de marqueur visuel permettant de distinguer un vaisseau sanguin d'un autre.

Divers travaux ont déjà abordé le problème du positionnement dans l'espace. Les méthodes d'aide à l'exploration [SCP95,ENK97,SS02] reproduisent le monde environnant, à différentes échelles, ou points de vue, pour aider l'utilisateur à se repérer plus facilement. On trouvera un recensement plus complet dans [Ste06, ET07], nous ne présentons ici que quelques unes de ces méthodes.

Les Worldlets [ENK97] sont des repères visuels sous forme de photographies 3D d'une scène. Cette méthode à pour avantage, à l'instar d'une image figée ou de texte, d'aider l'utilisateur à mieux percevoir l'environnement entourant un objet d'intérêt ou encore de lui permettre de l'observer depuis plusieurs endroits. Si les Worldlets sont particulièrement bien adaptés aux visites virtuelles en environnement urbain, dans notre cas, avec un environnement aussi peu naturel, l'absence de particularité locale rend les Worldlets peu intéressants. En effet, une photographie d'un vaisseau ou d'un embranchement, ne permet pas de se localiser mieux dans le monde, car vu de près, rien ne ressemble plus à un vaisseau sanguin qu'un autre vaisseau.

La technique dite *Through the Lens* [SS02] est une méthode permettant à l'utilisateur de voir le monde depuis un autre point de vue. L'utilisateur dispose d'une version miniature du monde. Il peut l'utiliser pour planifier son déplacement dans le monde à l'échelle 1 avant de le valider. Cette méthode est elle même inspirée du principe du *World in Miniature* (WIM) [SCP95]. Le WIM est une représentation miniature du monde dans laquelle l'utilisateur peut manipuler les objets et lui-même (sa propre position dans la scène est considérée comme un objet), en manipulant leurs versions miniatures. La technique du *Through the Lens* vise à palier la désorientation qui peut être créée par le fait de manipuler indirectement sa vision du monde. Le WIM seul tel que défini au départ ne suffit pas ici : les problèmes d'occlusions restent présent.

A coté des problèmes de positionnement, nous nous sommes penchés sur le développement de méthodes de sélection d'un objet fin ou d'une petite zone dans un volume. Comme décrit dans [Bow99, Ger04], on peut classer les outils de désignation en trois types : la désignation par préhension, la désignation indirecte et la désignation par pointage.

Une désignation par pointage seul ne permet pas de répondre au problème, car il est impossible de pointer précisément une zone cachée ou trop éloignée. Des améliorations telles que les pointeurs *flexibles* ou *courbes* permettant de contourner certains obstacles, se révèlent ici vite dépassées

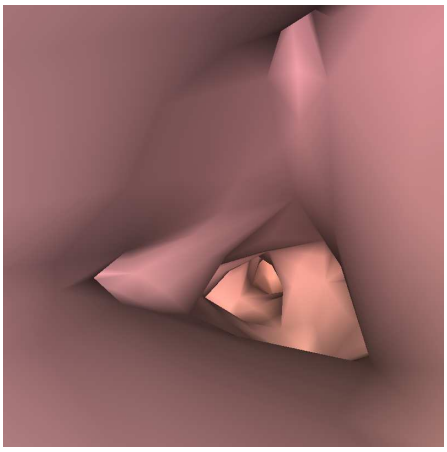


Figure 2: *Une vue de l'intérieur d'un vaisseau, lors d'une phase de navigation. Les zones claires signalent des embranchements accessibles.*

par le nombre important d'occlusions et l'enchevêtrement des vaisseaux.

La désignation par préhension n'est pas pertinente non plus, dans ce cadre, car la main de l'utilisateur, affichée dans l'environnement de réalité virtuelle crée un phénomène d'occlusion trop important. Elle empêcherai une sélection précise sur une miniature et poserai les même problèmes qu'une désignation par pointage à l'échelle 1. Nous avons donc choisi de développer une méthode de sélection volumique originale couplant la désignation par pointage et la désignation indirecte – basée sur notre implantation du WIM–.

Dans la littérature, les méthodes de désignation de ne préoccupent généralement pas de la notion de profondeur. Il s'agit en général, comme dans [TD97], d'effectuer un lancer de rayon et de désigner le premier objet d'intérêt rencontré. Cette méthode ne s'adapte pas aux objets à densité élevée, ne possédant pas de marqueurs d'intérêts particuliers.

Certaines méthodes ont été développées afin de permettre la sélection en environnement dense comme le *Bubble Cursor* [GB05] ou le *Splatter* [RRC*06]. La technique du *Bubble Cursor* sélectionne l'élément le plus proche du curseur. Dans la technique originale tous les objets sont affichés en permanence, comme nous y reviendrons plus tard la densité d'éléments sélectionnables de nos réseaux vasculaires est vraiment importante et l'affichage de tous ces éléments créent un phénomène d'occlusion. C'est pourquoi cette solution est peu satisfaisante dans notre situation. La technique du *Splatter explose* le volume original en sous parties ne se superposant pas. Dans notre cas la majorité des cellules que nous considérons sont de formes similaires et l'information de la position dans l'espace est celle qui nous importe : effectuer un éclatement nous ferait perdre cette notion.

Quatre autres méthodes de désignation dans des environnements denses ainsi qu'une évaluation de celles-ci sont décrites dans [GB06]. Il s'agit des *depth ray*, *lock ray*, *flower ray* et *smart ray*. Le *depth ray* permet de choisir l'objet à désigner selon la direction de la main de l'utilisateur et sa distance ; l'objet sélectionnable étant mis en valeur par une couleur différente. Le *lock ray* suit le même principe en effectuant cette sélection en deux étapes, empêchant ainsi d'avoir des imprécisions sur la direction lorsque l'on souhaite définir une distance. Le *flower ray* réparti tous les objets désignés par un rayon dans une première étape sous la forme d'un menu où les éléments ne se superposent pas. Le *smart ray* utilise la distance entre le rayon de sélection et le centre de chacun des objets en intersection et désigne l'objet dont le centre et le plus proche du rayon.

Le depth ray, le lock ray et le smart ray, possèdent les mêmes inconvénients que le Bubble Cursor de notre point de vue et ne résolvent pas les problèmes d'occlusions. Le flower ray s'apparente à la technique du Splatter et supprime l'information de la position géométrique.

Dans cet article, nous n'aborderons pas les problématiques liées à la manipulation ou au contrôle d'application. D'un coté, le vaisseau sanguin est considéré dans le cadre de la navigation comme étant rigide et ne nécessite donc pas d'interface ou d'outils spécifiques. D'autres part, les outils de manipulation, dans le contexte de la simulation d'angioscopie, sont plutôt liés aux instruments chirurgicaux permettant de diriger la caméra et s'écartent de notre problématique de navigation. Enfin, le contrôle d'application, en dehors de la gestion des déplacements, n'ouvre pas de problème spécifique.

Cet article est organisé comme suit : dans la deuxième section nous présentons informellement le modèle utilisé pour représenter les réseaux ainsi que divers algorithmes de parcours que nous avons implantés. Dans la troisième section nous abordons les aides au positionnement dans l'espace, notamment la nouvelle version du WIM proposée. La quatrième section donne un aperçu des méthodes d'interactions fournies dans l'application de réalité virtuelle pour la navigation et la désignation. Enfin la dernière section propose une discussion des résultats ainsi que quelques perspectives pour la suite de ce travail.

2. Modélisation des réseaux et navigation

Nous présentons dans cette partie la structure utilisée pour effectuer la modélisation des vaisseaux sanguins, ainsi que les différentes méthodes implantées afin d'effectuer un parcours dans ces derniers. De nombreuses méthodes de calcul de parcours ont déjà été proposées, nous n'effectuons pas l'état de l'art complet de ces méthodes ici, mais développons des notions utilisées pour mettre en place une navigation interactive et une navigation automatique.

2.1. Modèle volumique

La modélisation de notre réseau vasculaire est faites à partir des cartes combinatoires [Lie91] de dimension 3 et réalise une subdivision volumique de l'espace dans lequel la navigation a lieu. Les modèles sont basés sur une reconstruction volumique des vaisseaux à partir de données médicales. Un réseau vasculaire correspond donc à un ensemble de cellules topologiques, comme présenté sur la figure 3, cousues les unes aux autres par des relations d'adjacences. Chacune de ces cellules correspond géométriquement à un polyèdre convexe. On utilise les relations topologiques entre les différentes cellules pour pouvoir effectuer notre navigation (c'est à dire passer de cellules en cellules).

Bien que ce soit difficilement visible sur une image, il faut noter que les vaisseaux les plus épais et les zones d'embranchements sont subdivisés également dans leur épaisseur. Ainsi il existe des cellules au centre des vaisseaux et des cellules aux bords de ceux-ci (c'est à dire en contact avec la surface rendue sur l'image 3).

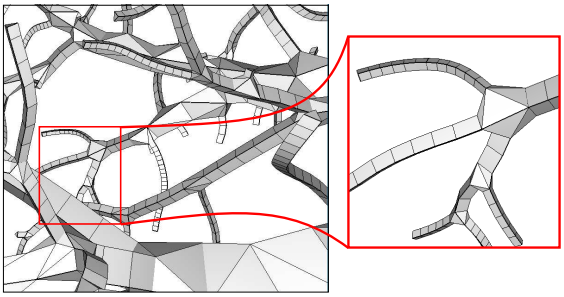


Figure 3: *Un vaisseau sanguin vu comme un ensemble de cellules adjacentes ; seuls les polyèdres du bord sont affichés ici, pour des raisons de visibilité.*

2.2. Calcul de parcours

Un parcours est défini dans notre simulation comme un ensemble continu, depuis une position de départ jusqu'à une position finale répondant à des critères d'admissibilité. Cet ensemble continu peut être défini comme une suite de cellules polyédriques adjacentes, ou comme une suite de positions dans l'espace à l'intérieur de ces polyèdres. Le critère d'admissibilité principal intervenant dans toutes nos navigations est le fait que le chemin calculé doit être sans collisions avec le bord (la surface entourant l'ensemble des polyèdres). Dans certains cas, on peut définir d'autres critères de type orthogonalité du chemin, ou distance par rapport à la paroi.

2.2.1. Parcours automatique

La navigation automatique permet de se déplacer de manière dirigée après avoir choisit une destination. Nous avons choisit d'utiliser pour ces parcours le principe du parcours A* [HNR68]. La structure de notre réseau veineux nous permet de construire un arbre de cellules adjacentes : tant que la destination n'est pas atteinte on développe les cellules connexes à la cellule ayant le plus de chance de nous rapprocher de notre but en empruntant le plus court chemin. Le choix de la cellule à développer est déterminé selon une heuristique basée sur la distance euclidienne jusqu'au point d'arrivée et la distance préalablement parcourue. En modifiant l'heuristique, on peut obtenir un chemin plus ou moins orthogonal ou encore calculer un chemin longeant les parois.

2.2.2. Parcours interactif

La navigation interactive consiste à laisser l'utilisateur choisir une direction avec un outil à 6 degrés de liberté de manière libre. Nous basons cette navigation sur le parcours direct [DPT02], c'est à dire un parcours en ligne droite ; à chaque changement de direction pointée, un parcours est recalculé. Avec ce parcours, la détection d'une collision résulte en l'impossibilité d'avancer. La structure en subdivision volumique nous permet ici d'effectuer des calculs de collisions locaux à chaque cellules, réduisant ainsi les temps de calculs.

La caméra est considérée comme un point se déplaçant avec un volume sphérique. Ce volume peut être utilisé dans deux optiques : pour une simulation, on peut prendre en compte la taille de l'outil chirurgical, pour une exploration, on peut empêcher la caméra de se rapprocher trop près d'une paroi et éviter ainsi des vues inintéressantes.

Nous ne développons pas ici les techniques de désignations de directions, diverses études traitant déjà de ce sujet. On pourra consulter [BKH97, Ste06] pour une évaluation de certaines méthodes ou [MFPBS97, SB05] pour des propositions d'indications de directions basées sur l'utilisation bimanuelles de gants de données. Nous avons opté pour la méthode dites de la soucoupe volante – *Flying Saucer* –, où l'utilisateur pointe une direction avec un outil d'interaction tout en ayant la possibilité de regarder dans une autre.

3. Se repérer dans l'espace

Certains problèmes d'orientation apparaissent lorsque l'on navigue dans un réseau. Il y a la difficulté de l'utilisateur à repérer à quel endroit du système vasculaire il se situe, car toutes les zones sont sensiblement identiques dans la vue immersive du réseau.

Le mal du simulateur lors d'une navigation automatique n'est pas à négliger non plus. Ce phénomène s'explique par le fait que l'utilisateur ne contrôle pas les changements de directions. Il n'arrive donc pas à anticiper ses mouvements et se retrouve désorienté quand il effectue un virage inattendu. Cet effet est accentué par le fait qu'il ne soit pas possible de distinguer clairement les notions de verticale ou d'horizontale ni de gauche ou de droite, l'environnement n'ayant pas de sens à proprement parlé.

Nous avons mis en place des outils pour aider l'utilisateur à se repérer dans l'espace et pour lui permettre d'anticiper les changements d'orientation dans le cas d'une navigation automatique. Ces outils sont le WIM, le fil d'Ariane et la ligne de visée adaptative, on peut avoir un aperçu des deux premiers outils cités sur la figure 7.

3.1. WIM

Le WIM que nous mettons en place permet à l'utilisateur d'avoir une meilleure vision du monde dans sa globalité. La vue *principale* est une vue intérieure du vaisseau –cf figure 2–, le WIM est asservi à cette vue en complément d'information. Nous proposons un affichage en mode fil de fer pour résoudre partiellement le problème de l'occlusion, nous reviendrons dans la section 4.1 sur ce problème.

Contrairement au cas du WIM tel que décrit dans [SCP95], nous ne donnons pas la possibilité à l'utilisateur de modifier sa position directement en manipulant sa représentation – le cône de positionnement – car l'environnement étant étroit, cette tâche, à petite échelle s'avère laborieuse.

Par contre, nous avons couplé le WIM avec un outil de

type Flystick permettant de le manipuler aisément. Notre WIM peut être déplacé ou orienté suivant 6 degrés de liberté. Cela permet notamment de positionner les zones d'intérêt vers soi. Il est possible de changer l'échelle dans le cas où l'utilisateur souhaite observer une zone plus en détail avant de s'y déplacer. Un cône de positionnement est également intégré dans le WIM et sert à représenter la position et l'orientation courante de l'utilisateur à l'intérieur du réseau veineux.

Afin de ne pas obstruer inutilement la vue de l'utilisateur lorsqu'il souhaite observer la paroi du vaisseau sanguin, nous avons rajouté la possibilité de faire apparaître ou disparaître intégralement le WIM du champ visuel.

3.2. Fil d'Ariane

Le fil d'Ariane est un indicateur du parcours précalculé dans le cadre d'un parcours automatique. Il apparaît à deux échelles, premièrement au niveau de la vue immersive et deuxièmement il est reproduit à l'échelle correspondante dans le WIM. Il a deux avantages. D'une part, il améliore l'anticipation des virages et donc réduit le mal du simulateur abordé précédemment. D'autre part, il permet d'estimer la distance à parcourir pour arriver à la cellule destination en l'observant sur le WIM avant de l'arpenter.

3.3. Ligne de visée adaptative

La ligne de visée, de la même façon que le fil d'Ariane, va permettre à l'utilisateur de visualiser son parcours. Cette ligne de visée est employée dans le cadre de la navigation interactive, pour indiquer à l'utilisateur, par une demi-droite, l'endroit vers lequel il se dirige. Elle est adaptative dans le sens où sa longueur varie en fonction de la distance au bord.

Cette demi-droite s'arrête lors d'une collision avec la paroi. Il est alors plus facile d'estimer la distance entre la position courante de la caméra et le bord de l'environnement. Le fait de pouvoir prévoir sa distance par rapport au bord aide l'utilisateur à corriger sa trajectoire lorsqu'il se rapproche trop près d'une paroi. Cela permet également de mieux percevoir la direction pointée, en l'associant à la profondeur *accessible* dans cette direction.

4. Sélection de cellules

Il n'existe pas, à notre connaissance, de méthodes satisfaisantes de sélection de cellules adaptée à notre cadre applicatif. Comme l'illustre la figure 4, nos réseaux vasculaires sont composés d'un grand nombre de cellules réparties de manière dense et complexe. La sélection dans un volume que nous souhaitons utiliser doit pouvoir nous permettre de choisir une boule dans un tas, sans ôter celle qui se trouvent devant. Notre méthode aborde ce problème de deux manières.

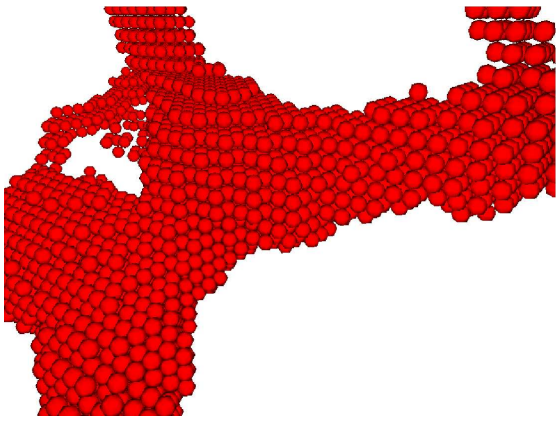


Figure 4: *Représentation des cellules d'un vaisseau sanguin.*

Nous souhaitons dans un premier temps simplifier la complexité de cette tâche en effectuant un affichage partiel de notre objet, en mettant en place un plan de coupe. Dans un deuxième temps, nous créons une métaphore que nous appelons la métaphore de la chandelle, afin de nous permettre de sélectionner de manière intuitive une cellule de notre vaisseau sanguin de manière précise.

4.1. Plan de coupe

Comme le montre la figure 5 sur l'image de gauche représentant le WIM, il peut s'avérer parfois difficile de voir clairement une zone à cause de la présence de nombreuses veinules. Pour palier à ce problème, un plan de coupe manipulable a été implanté. Celui-ci permet d'afficher une vue partielle du système dans le WIM. Nous proposons à l'utilisateur de définir un plan de coupe, qu'il peut déplacer, afin de se focaliser sur une zone précise du WIM.

Le fait d'utiliser un plan de coupe permet d'avoir une meilleure notion de la profondeur en nous projetant uniquement sur deux dimensions au niveau du plan de coupe. Lorsque le plan de coupe, dans un sens ou un autre, cache ou montre entièrement la miniature, il disparaît et permet ainsi de recouvrir l'affichage standard.

Cette méthode est plus efficace lorsque l'objet est de taille suffisamment faible pour ne pas obliger l'utilisateur à effectuer de trop grand gestes, ces derniers fatiguant rapidement dans un environnement de réalité virtuelle.

4.2. Métaphore de la chandelle

La métaphore de la chandelle, illustré par la figure 6, est un outil de sélection d'objet à l'intérieur d'un volume. Il suit en quelque sorte le principe du FishEye [WCS01, CB04] sans déformer le maillage originel sur lequel on l'applique, et le principe du *depth ray* décrit dans la section 1.

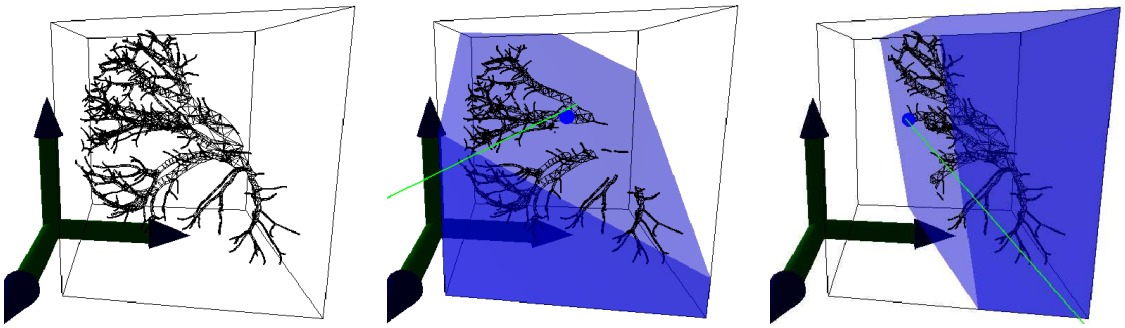


Figure 5: *Utilisation du plan de coupe.*

L'utilisateur manipule un rayon laser au bout duquel est placé une *chandelle* virtuelle. Le rayon laser permet de choisir une direction et une profondeur pour placer cette dernière. La *chandelle* éclaire les cellules visées, dans une sphère d'influence de rayon réglable. La *lumière* fait apparaître de couleur plus vive et de taille plus grandes les cellules proches de la *mèche*.

La perception de la profondeur est améliorée par cet éclairage qui met en évidence la profondeur relative des cellules. La perception de l'espace environnant est également amélioré, car la zone d'influence permet l'évaluation des distances entre les différentes parois. Les cellules intersectées directement par le laser sont mises en valeur à l'aide d'une couleur différente pour pouvoir mieux évaluer la direction pointée.

Afin de faciliter la sélection de l'utilisateur il est possible de désigner une cellule sans l'intersecter directement avec le laser. La cellule la plus proche de la « mèche de la chandelle » est alors désignée.

5. Conclusion

Les résultats actuels nous permettent de naviguer interactivement dans un objet concave, non déformable, mais d'une grande complexité, et ce dans un environnement de type Workbench avec vision stéréographique (environ 120Hz). L'application est couplée avec un outil d'interaction à six degré de liberté de type Flystick. L'objet déplacé est actuellement un point possédant un volume, son déplacement étant simplement arrêté lorsqu'une collision est détectée : invitant ainsi l'utilisateur à se diriger dans une autre direction.

Les apports de la visualisation stéréoscopique apparaissent lorsque la profondeur de champ de vue est importante. Elle permet en effet d'avoir une meilleur perception de son environnement. La réalité virtuelle apporte, à notre sens, également une aide pour la sélection volumique, en facilitant l'interaction de l'utilisateur.

La métaphore de la chandelle permet de sélectionner des cellules dans un espace volumique, le couplage avec le plan de coupe résolvant les problèmes d'occlusions. Cette méthode de sélection donne une meilleure notion de distance entre les différentes cellules que l'on souhaite désigner.

On peut utiliser notre application, si l'interactivité n'est pas demandée, comme un outil d'exploration plutôt que de simulation. Les outils de sélection développés permettent de choisir un chemin et d'explorer ainsi l'espace désigné.

Il est souhaitable d'effectuer un comparatif entre les méthodes de sélection que nous avons mises en place et les autres solutions existantes. On pourra notamment effectuer des tests sur des utilisateurs pour mesurer la vitesse de sélection, le taux d'erreur et l'aisance d'apprentissage de celles-ci. Nous souhaitons également mesurer l'influence des déformations du maillage d'origine, sur la perception, en utilisant la technique du FishEye. On cherchera ainsi à savoir si cela perturbe ou aide l'utilisateur dans son repérage.

Au niveau de l'aide à la visualisation on pourra effectuer des tests mesurant la sensation de malaise selon divers paramètres de vitesse, et la présence ou non d'indices visuels visant à la diminuer.

Références

[BKH97] BOWMAN D. A., KOLLER D., HODGES L. F.: Travel in immersive virtual environments: an evaluation of viewpoint motion control techniques. *IEEE Proceedings of VRAIS'97*, 7 (1997), 45–52.

[Bow99] BOWMAN D. A.: *Interaction Techniques For Common Tasks In Immersive Virtual Environments - Design, Evaluation, and Application.* PhD thesis, Georgia Institute of Technology, 1999.

[CB04] COHEN M., BRODLIE K.: Focus and context for volume visualization. *Theory and Practice of Computer Graphics* (2004), 32–39.

[DPT02] DEVILLERS O., PION S., TEILLAUD M.: Walking in a triangulation. *International Journal of Foundations of Computer Science 13*, 2 (2002), 181–199.

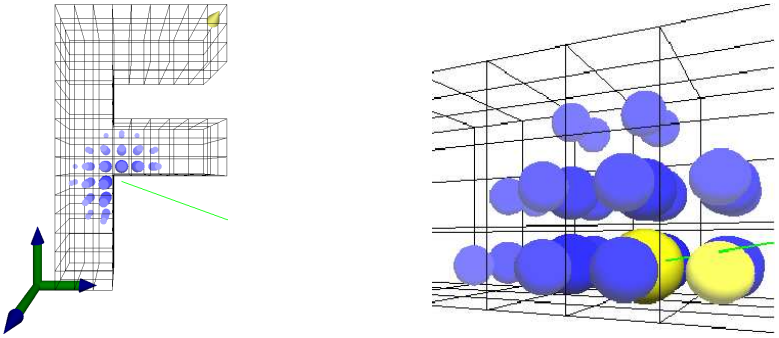


Figure 6: *Métaphore de la chandelle.*

[ENK97] ELVINS T. T., NADEAU D. R., KIRSH D.: Worldlets 3d thumbnails for wayfinding in virtual environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1997), ACM Press, pp. 21–30.

[ET07] ELMQVIST N., TSIGAS P.: A taxonomy of 3d occlusion management techniques. In *Proceedings of the IEEE Conference on Virtual Reality 2007* (2007), pp. 51–58.

[GB05] GROSSMAN T., BALAKRISHNAN R.: The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2005), ACM Press, pp. 281–290.

[GB06] GROSSMAN T., BALAKRISHNAN R.: The design and evaluation of selection techniques for 3d volumetric displays. In *UIST '06: Proceedings of the 19th annual ACM symposium on User Interface Software and Technology* (New York, USA, 2006), ACM Press, pp. 3–12.

[Ger04] GERBER D.: *Interaction 3D sur un plan de travail virtuel : Application aux déformations de forme libre*. PhD thesis, Université Louis Pasteur, 2004.

[HNR68] HART P., NILSSON N., RAPHAEL B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics volume 4* (1968).

[Lie91] LIENHARDT P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Comput. Aided Des. 23*, 1 (1991), 59–82.

[MFPBS97] MINE M. R., FREDERICK P. BROOKS J., SEQUIN C. H.: Moving objects in space: exploiting proprioception in virtual-environment interaction. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 19–26.

[RRC*06] RAMOS G., ROBERTSON G., CZERWINSKI M., TAN D., BAUDISCH P., HINCKLEY K., AGRAWALA M.: Tumble! splat! helping users access and manipulate occluded content in 2d drawings. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces* (New York, USA, 2006), ACM Press, pp. 428–435.

[SB05] STERNBERGER L., BECHMANN D.: Deformable ray-casting interaction technique. In *IEEE Young Virtual Reality Conference* (february 2005).

[SCP95] STOAKLEY R., CONWAY M. J., PAUSCH R.: Virtual reality on a wim: interactive worlds in miniature. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, USA, 1995), ACM Press/Addison-Wesley Publishing Co., pp. 265–272.

[SS02] STOEV S. L., SCHMALSTIEG D.: Application and taxonomy of through-the-lens techniques. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, USA, 2002), ACM Press, pp. 57–64.

[Ste06] STERNBERGER L.: *Interaction en Réalité Virtuelle*. PhD thesis, Université Louis Pasteur, 2006.

[TD97] TÖNNIES K. D., DERZ C.: Volume rendering for interactive 3-d segmentation. In *Proceedings of the SPIE (Medical Imaging 1997)* (1997), vol. 3031, pp. 602–609.

[WCS01] WINCH D., CALDER P., SMITH R.: Solving the occlusion problem for three-dimensional distortion-oriented displays. *User Interface Conference, 2001* (2001), 108–115.

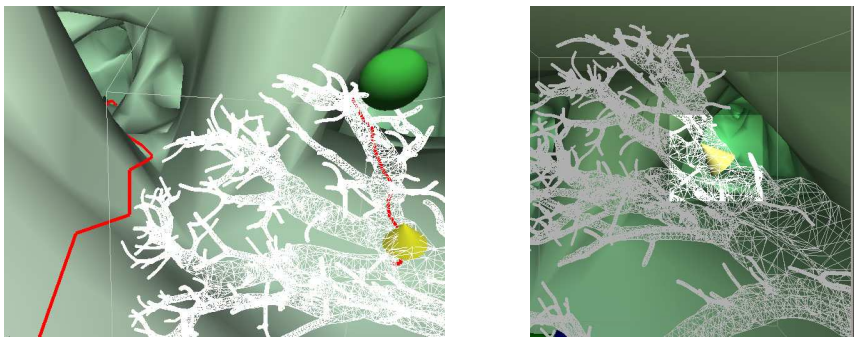


Figure 7: *Fil d'Ariane, cône de positionnement lors d'une navigation dans un vaisseau sanguin en présence d'un WIM.*

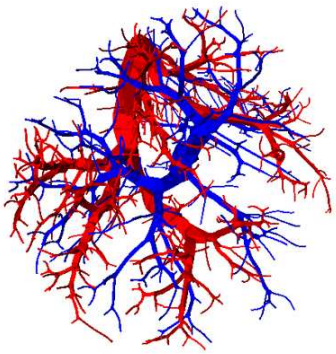


Figure 8: *Deux parties du réseau vasculaire s'imbriquant.*

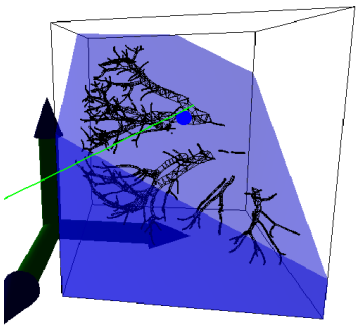


Figure 9: *Vue couleur d'un plan de coupe sur un WIM.*

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Adaptation dynamique de mouvements aériens

Ludovic Hoyet [1], Franck Multon [2], Taku Komura[3], Richard Kulpa[2]

[1]INSA de Rennes et Projet Bunraku, IRISA
[2]M2S, Université de Rennes 2 et Projet Bunraku, IRISA
[3]Université d'Edimbourg

Abstract

Cet article traite de la validité dynamique de mouvements capturés pour des applications interactives. Il se focalise sur la correction de mouvements aériens, tels que ceux réalisés par les gymnastes, où nous faisons l'hypothèse que les forces extérieures sont limitées à la gravité. La dynamique du mouvement est alors entièrement définie par différentes lois mécaniques et ne peut donc pas être négligée. Nous nous inspirons des méthodes itératives résolvant les contraintes cinématiques, pour proposer une approche corrigeant les mouvements aériens. En prenant en compte les données morphologiques propres au personnage (poids, taille et inerties), nous sommes capables de corriger le mouvement capturé.

This paper adresses the problem of the physical correctness of captured motions, in interactive applications. It focus on correction of aerial motions, such as gymnastic ones, assuming no other forces than gravity. Then, dynamic of the motion is completly defined by mecanical laws and can't be neglected. Our method is inspired from kinematic constraints solvers, in order to propose a way to correct aerial motions. Taking into account character's own morphology (mass, inertia and size), we are able to correct the captured motion.

1. Introduction

Les systèmes de capture de mouvements fournissent des trajectoires, généralement enregistrées pour la morphologie du sujet d'origine, qui nécessitent d'être retraitées avant de les appliquer à des nouveaux personnages. Il est aussi parfois nécessaire de déformer ces mouvements, par exemple pour modifier la position d'un objet à atteindre, ce qui nécessite d'utiliser différentes contraintes cinématiques. Il est de même possible de combiner différents mouvements pour réaliser des tâches complexes. Dans tous les cas, les mouvements obtenus perdent généralement de leur validité et de leur réalisme dynamique [SH05].

Une gamme de mouvements très sensibles aux paramètres dynamiques est celle des mouvements aériens, où la seule force s'exerçant sur le corps est la gravité. Dans ce type de mouvements, la modification des paramètres anthropométriques (masses, inerties, taille des segments) entraîne un modification complète du mouvement. De tels effets ont été étudiés en biomécanique pour les mouvements gymniques [Yea93]. Par exemple, il a été démontré qu'il est possible d'initier des vrilles au cours d'une phase aérienne, alors que le corps n'était préparé que pour un simple salto au décollage. Comme le montre la figure 4, un mouvement du bras au cours d'une phase aérienne peut en effet entraîner la création d'une vrille. La méthode que nous proposons ici vise à produire et à vérifier de tels comportements. Elle vise des applications temps réelles telles que les jeux vidéos, ou encore les systèmes d'aide à l'entraînement de sportifs (où les athlètes peuvent interagir avec des personnages virtuels pour appréhender des figures dangereuses).

Plusieurs méthodes ont déjà été proposées pour gérer de telles propriétés dynamiques dans des applications ne nécessitant pas d'interactions temps-réel. Ces méthodes sont basées sur la simulation de la configuration du corps entier [Hod95] ou sur de l'optimisation [WK88]. Des méthodes consistant à rechercher un comportement réaliste dans une base de données de mouvements existent aussi mais, bien que plus rapides, restent limitées à des perturbations données, de très courte durée [ZMCF05]. Durant les phases aériennes, il est nécessaire de corriger le mouvement à chaque pas de temps en utilisant les données anthropométriques du

personnage : deux personnages de morphologie différente ne réaliseront pas exactement le même mouvement.

Cet article pose le problème de l'adaptation temps réelle de mouvements aériens capturés, tels que des figures gymniques. Nous y proposons une méthode, développée pour des applications interactives temps réelles, permettant de corriger pas à pas les propriétés dynamiques de ce type de mouvements.

2. Travaux existants

Les méthodes permettant de prendre en compte la dynamique du mouvement d'humains virtuels peuvent être séparées en deux principales familles : celles générant des mouvements à partir de lois et de paramètres et celles adaptant les trajectoires existantes.

Le problème de la simulation de mouvements dynamiquement valides a tout d'abord été traité par la création de contrôleurs [vdPF93, Hod95, HP97, FvdPT01] qui génèrent les valeurs des angles aux articulations du squelette, en fonction de tâches précises et de l'état du système. Cependant, un contrôleur reste spécifique à une tâche donnée et l'obtention d'un mouvement visuellement naturel demande beaucoup de travail.

Une alternative à la création de contrôleurs consiste à capturer des mouvements et à les réutiliser, en conservant donc leur réalisme. L'adaptation de tels mouvements à de nouveaux personnages et à de nouvelles contraintes cinématiques a déjà été longuement traité [Gle98, CK00, Gle97, LS99, KMA05]. Quelques améliorations ont été proposées, visant à préserver la validité dynamique de ces mouvements, par exemple en contrôlant le moment s'exerçant aux articulations du squelette [ZH99, ZH02, LHP06]. McCann et coll. [MPS06] proposent quand à eux de modifier l'échelle de temps du mouvement tout en prenant en compte certaines contraintes dynamiques. Leurs bonnes performances nécessitent alors le pré-calcul des principaux paramètres dynamiques. Cependant, cette méthode est limitée à la dilatation ou à la compression de la vitesse d'exécution du mouvement et ne permet pas d'ajuster la posture du personnage. Le filtrage dynamique [TSK02, YN03, TK05] consiste à prédire une posture physiquement correcte à partir des précédentes postures, puis à filtrer le mouvement en prenant en compte à la fois les postures capturées et les postures prédites. Cette approche permet d'adapter un mouvement capturé à un personnage ayant une morphologie différente. Cependant, elle nécessite de définir différents paramètres dynamiques, fonctions du mouvement utilisé, qui se doivent d'être correctement choisis au risque de ne pas obtenir un mouvement naturel.

Récemment, plusieurs travaux se sont intéressés à l'utilisation conjointe de la simulation et de bases de données de mouvements capturés, dans le but de faire réagir de façon naturelle des personnages soumis à des collisions ou à des chocs [ZMCF05, AFO05, KHL05]. L'idée principale est de laisser la simulation diriger le squelette humain comme une hiérarchie passive de segments rigides, tout en cherchant simultanément dans la base de donnée un mouvement de réaction compatible à mélanger à la simulation. La simulation est ici efficace pour calculer un début de réaction réaliste au choc et l'utilisation d'un mouvement capturé assure que la fin du mouvement est naturelle et contrôlable. Cependant, bien que ces méthodes soient efficaces pour gérer des réactions à des chocs, elles ne peuvent être utilisées pour adapter un mouvement avec des contraintes dynamiques continues.

Les contraintes spatio-temporelles ont quand à elle été introduites dans le but de contrôler différentes figures au travers d'un processus d'optimisation, prenant en compte à la fois des contraintes cinématiques et dynamiques [WK88, Coh92, LGC94, RGBC96, SHP04]. Ce processus est alors dirigé par la minimisation d'une fonction de coût. En conséquent, ces méthodes sont très sensibles aux minimaux locaux et coûteuses en temps de calculs. Certains auteurs ont alors proposés de contrôler de simples contraintes dynamiques (telles que le moment cinétique [LP02] ou les forces extérieures s'exerçant sur le squelette [FP03]), dans le but d'accélérer l'optimisation. Bien que les temps de calculs soient réduits de manière significative, ces méthodes sont principalement utilisées en pré-processus pour enrichir une base de données existante contenant des mouvements dynamiquement valides. Cependant, ces bases de données ne sont utilisables que pour un squelette et une morphologie donnés. L'utilisation de personnages différents implique donc la création de plusieurs bases de données, chacune comprenant la gamme de mouvements pour un unique personnage. De plus, dans les applications interactives, un surpoids peut être associé à un segment du corps pour représenter des objets externes avec une masse, le tout de façon imprévue. Notre méthode n'est pas conçue pour réaliser de grandes modifications du mouvement d'origine, mais simplement pour corriger le mouvement en fonction des paramètres morphologiques des personnages.

L'idée de contrôler interactivement différentes contraintes dynamiques, au lieu du moment aux articulations du squelette, a aussi été explorée sans avoir recours à des processus d'optimisation. Par exemple, il est possible de normer les forces qui s'appliquent sur le mouvement capturé dans le but de résoudre de nouvelles contraintes [PBM00]. Le mouvement est alors divisé en différentes séquences, représentant différentes poses. Shin et coll. [SKG03] proposent par exemple de vérifier que la projection du point de moment nul (Zero Moment Point) reste à l'intérieur du polygone de sustentation au cours des phases de contact avec le sol. Lors des phases aériennes, leur méthode assure que le centre de masse suit une trajectoire parabolique d'accélération égale à la gravité. La posture est alors adaptée en utilisant des cartes de déplacements, permettant d'ajouter des petits déplacements à la position du personnage pour corriger la dynamique. Notre méthode s'apparente à cette catégorie. Dans

cet article, nous nous focalisons sur l'étude de mouvements aériens en contrôlant la position du centre de masse et le moment cinétique. L'utilisation de cartes de déplacement nécessite cependant un processus itératif et la connaissance totale du mouvement, ce qui est incompatible avec le cadre interactif dans lequel nous nous plaçons. L'idée principale est ici de résoudre les contraintes dynamiques à chaque pas de temps, l'environnement interactif pouvant introduire des changements imprévus suite à l'intervention de l'utilisateur ou d'un personnage autonome.

3. Contraintes liées à la dynamique

Les postures d'humanoïdes virtuels sont généralement représentées par la position de la racine du squelette et par les angles aux articulations. Dans un tel cas, il est nécessaire d'avoir recours à des contraintes spatio-temporelles ou à de la cinématique inverse pour éditer un mouvement ou pour l'adapter à une autre morphologie que celle du sujet d'origine [Gle98]. Récemment, une représentation du squelette indépendante de la morphologie a été proposée pour éviter d'avoir recours à ces coûteux algorithmes [KMA05]. Cette représentation couple des données cartésiennes et angulaires, normalisées pour obtenir un squelette adimensionnel. Par exemple, un bras est représenté par une ligne joignant l'épaule au coude et par l'angle représentant la direction dans laquelle le coude est situé. Avec une telle représentation, il est simple d'appliquer un mouvement à un nouveau personnage. Dans la suite de cette article, nous utilisons cette représentation pour associer un mouvement à un nouveau personnage et pour résoudre rapidement diverses contraintes cinématiques. Voyons maintenant qu'elles sont les propriétés dynamiques à prendre en compte lors des phases aériennes.

En phase aérienne, les lois de la dynamique énoncent que la position du centre de masse du personnage doit suivre une trajectoire parabolique. De la même façon, le moment cinétique doit rester constant jusqu'à la réception :

$$L = \sum_{i=1}^{n} I_i \omega_i + m_i GG_i \times \dot{GG_i} = cste \quad (1)$$

où n est le nombre de segments du squelette, m_i, I_i, ω_i et G_i sont la masse, l'inertie, la vitesse angulaire et le centre de masse du segment i. Ces paramètres dynamiques ont été largement étudiés dans le domaine de la biomécanique. Des tables anthropométriques, renseignant les différentes proportions de chaque segment du corps humain, sont donc utilisées pour les évaluer [Lev96].

La vitesse angulaire d'un segment i est donnée par :

$$\omega_i = R_w^i(\dot{\psi_i}, \dot{\theta_i}, \dot{\phi_i})^T \quad (2)$$

où $(\dot{\psi_i}, \dot{\theta_i}, \dot{\phi_i})$ sont les dérivées des trois angles d'euler représentant l'orientation du segment i dans le repère global, et R_w^i est la matrice de rotation du repère global vers le repère local.

Lorsque l'on modifie interactivement un mouvement, en modifiant la position d'un bras par exemple, ou si l'on applique un mouvement à un personnage différent du sujet d'origine, il y a perte de la validité dynamique du mouvement. Pour corriger cela, nous proposons deux méthodes prenant en compte les données morphologiques propres au personnage :

- Correction de la vitesse de rotation globale : l'inertie globale du personnage n'est pas modifiée. Il faut alors adapter la vitesse de rotation globale du corps pour conserver le moment cinétique constant (Section 4).
- Correction de la posture : il faut modifier l'inertie globale du personnage en corrigeant sa posture (Section 5)

Quelle que soit l'approche utilisée, la position du centre de masse (*COM*) du personnage est aussi affectée et ne suit plus une trajectoire parabolique. Soit ΔCOM l'écart entre la position actuelle du *COM* et celle donnée par l'équation parabolique. Pour corriger la position du *COM*, il suffit alors de translater la position de la racine du personnage de ΔCOM.

4. Correction de la vitesse de rotation globale

Lorsque l'on utilise des mouvements issus de capture, ceux-ci peuvent être rejoués par des personnages ayant une morphologie différente de celle du sujet d'origine. Par exemple, un personnage ayant une masse plus importante que celle du sujet d'origine aura aussi une inertie plus importante et, par conséquent, une vitesse de rotation plus faible pour le même saut. Le but de la correction de la vitesse de rotation globale est donc de prendre en compte la morphologie propre au personnage qui joue le mouvement et d'adapter la vitesse de rotation tout en conservant la posture caractérisant le saut.

Pour corriger la vitesse de rotation, nous calculons tout d'abord le moment cinétique initial L_0, au premier instant suivant le décollage. Le moment cinétique est censé rester constant au cours de la phase aérienne et être égal à L_0. Cependant, à cause de la morphologie différente des personnages utilisés et de différents autres facteurs, la valeur du moment cinétique varie au cours des phases aériennes (courbes continues de la figure 2). Dans cette section, nous souhaitons conserver la gestuelle du mouvement et, par conséquent, conserver les postures successives. Ainsi, nous adaptons la vitesse de rotation globale ω_1 de la racine du personnage pour corriger la valeur du moment cinétique. Chaque autre segment i possède aussi une vitesse de rotation ω_i, qu'il est possible d'exprimer en fonction d'ω_1 et de la vitesse de rotation $\omega_{i/1}$ du segment i par rapport à la racine du personnage :

$$\omega_i = \omega_1 + \omega_{i/1} \quad (3)$$

Soit q le vecteur contenant tous les degrés de liberté du personnage et $q_i = (\psi_i, \theta_i, \phi_i)$ le vecteur contenant les trois

degrés de liberté du segment i. On a alors $q = \{q_i\}_{i=1..N}$. L'équation 1 peut alors s'écrire :

$$L = \sum_{i=1}^{n} I_i(\omega_1 + \omega_{i/1}) + f_i(q,\dot{q}) \quad (4)$$

où $f_i(q,\dot{q}) = m_i GG_i \times \dot{GG_i}$

Comme nous ne souhaitons modifier que la vitesse de rotation ω_1 et conserver la posture d'origine, les vecteurs $q_{i\neq 1}$ et leur dérivée sont donc connus pour tous les segments, excepté pour le tronc : $f_i(q,\dot{q})$ peut alors s'exprimer uniquement en fonction des inconnues q_1 et $\dot{q_1}$. La posture au temps t étant correcte, nous souhaitons modifier la vitesse de rotation ω_1 entre t et $t+\Delta t$ pour obtenir une posture correcte à $t+\Delta t$. L'orientation q_1 au temps t étant connue, la seule inconnue de la fonction $f_i(q,\dot{q})$ est donc $\dot{q_1}$. ω_1 pouvant aussi s'exprimer en fonction de $\dot{q_1}$, l'équation 1 devient :

$$L - \sum_{i=1}^{n} I_i\omega_{i/1} = \sum_{i=1}^{n} I_i R_w^i \dot{q_1} + f_i(\dot{q_1}) \quad (5)$$

Rappelons que $w_{i/1}$ est connue dans cette équation puisqu'il s'agit de la rotation du segment i par rapport à la racine du personnage. Puisque nous ne souhaitons pas modifier la posture, $w_{i/1}$ est donc inchangée et peut être directement obtenue à partir du fichier.

Des temps de calculs les plus faibles possibles faisant partis de nos impératifs, nous avons proposé de linéariser la fonction f_i afin d'éviter le recours à des techniques d'optimisation :

$$\Delta f_i(\dot{q_1}) = J_i \Delta\dot{q_1} \quad (6)$$

où J_i est le jacobien associé aux trois angles de rotation du segment i, obtenu numériquement par la méthode des différences finies avec par exemple :

$$\frac{\partial f_{ix}}{\partial \psi_1} = \frac{f_i(\psi_1 + \delta_\psi, \theta_1, \phi_1) - f_i(\psi_1 - \delta_\psi, \theta_1, \phi_1)}{2\delta_\psi}$$

où δ_ψ est une petite variation appliquée à l'angle ψ. Il en est de même pour les trois angles.

L'équation 5 devient alors :

$$L = \left(\sum_{i=1}^{n} I_i\omega_{i/1} - J_i\hat{\dot{q}_i} + \widehat{f_i(\dot{q_1})}\right) + \left(\sum_{i=1}^{n} I_i R_w^i + J_i\right)\dot{q_1} \quad (7)$$

où $\widehat{f_i(\dot{q_1})}$ correspond à la valeur de la fonction f_i de la posture non corrigée et $\hat{\dot{q}_i}$ la valeur de $\dot{q_i}$ de la posture non corrigée. Puisque nous voulons calculer $\dot{q_1}$ permettant d'atteindre le moment cinétique L_0, nous avons :

$$\dot{q_1} = \left(\sum_{i=1}^{n} I_i R_w^i + J_i\right)^{-1}\left(L_0 - \sum_{i=1}^{n} I_i\omega_{i/1} - J_i\hat{\dot{q}_i} + \widehat{f_i(\dot{q_1})}\right) \quad (8)$$

Et donc :

$$q_1(t+\Delta t) = q_1(t) + \dot{q_1}\Delta t \quad (9)$$

Comme il y a plusieurs approximations mathématiques et numériques dans la méthode proposée, le fait d'utiliser directement les résultats de l'équation 9 peut faire apparaître de petites discontinuités dans le mouvement corrigé. Pour limiter cet effet, il est possible d'utiliser un filtre passe-bas. Comme nous nous plaçons dans le cadre d'animation temps-réelle interactive, il nous est nécessaire d'obtenir un compromis entre temps de calculs et précision. Calculer exactement la matrice jacobienne J_i entraîne des temps de calculs qui altéreraient énormément les performances de notre méthode. Le fait d'appliquer un filtre passe-bas est par conséquent moins coûteux, bien qu'entraînant une approximation des lois de la dynamique.

5. Correction de la posture

Pour garder la valeur du moment cinétique constante durant les phases aériennes, une seconde stratégie consiste à modifier la rotation propre aux segments du personnage. Cependant, il peut y avoir une infinité de configurations respectant la valeur L_0 du moment cinétique. De même que dans le domaine de la cinématique inverse, deux principales solutions s'offrent à nous. Une première consiste à faire intervenir chaque segment du squelette en leur associant un poids, chacun compensant une partie du moment cinétique. Cependant, cette méthode est très sensible aux poids choisis et peut engendrer des mouvements peu réalistes. La seconde solution consiste à modifier dans un certain ordre les différents segments du squelette, sans modifier l'orientation globale du personnage. Par exemple, il est possible de modifier la vitesse de rotation d'un bras pour diminuer l'écart entre le moment cinétique actuel et L_0. Si l'utilisation de ce bras n'est pas suffisante pour atteindre L_0, un ou plusieurs autres segments sont utilisés successivement. Cette dernière stratégie est celle utilisée par les algorithmes de type Cyclic Coordinate Descent [WC91]. Il peut cependant arriver qu'un segment se déplace excessivement pour respecter la contrainte, entraînant un mouvement non réaliste.

Le choix de la stratégie à utiliser étant important pour obtenir des mouvements réalistes, nous avons étudié comment les athlètes contrôlaient leur corps durant les phases aériennes. En sport, et particulièrement en gymnastique, il a été montré que les personnes entraînées utilisent principalement leurs bras pour contrôler le moment cinétique total [Yea93]. En effet, bien que les bras possèdent une masse et une inertie faibles, leur grande mobilité autorise des vitesses importantes, ce qui permet un bon contrôle de la rotation. De même, lorsque des personnes inexpérimentées essayent de réaliser des mouvements aériens complexes (vrilles, saltos, etc), ils réalisent naturellement de grands mouvements des bras pour essayer de contrôler leur rotation.

Par conséquent, nous avons choisi la seconde solution pour corriger le moment cinétique. Les bras sont tout d'abord sollicités, puis les jambes si nécessaire. Finalement, si le moment cinétique n'a pas pu être corrigé totalement,

nous adaptons la rotation du tronc par la méthode décrite dans la section 4. Cependant, si nous adaptons indépendamment les segments d'un membre, par exemple la cuisse et le mollet pour la jambe, nous risquons d'obtenir des mouvements non réalistes. L'articulation intermédiaire (coude ou genou) risque de subir des adaptations importantes conduisant à des postures irréalistes. Pour éviter ce problème, nous avons choisi d'adapter les segments par groupe en une seule opération. Un groupe est adapté en une seule fois, au lieu de réaliser indépendamment la rotation de chaque segment le composant, comme suggéré par [KMA05] pour l'adaptation des contraintes cinématiques et cinétiques.

Dans notre approche, nous considérons cinq groupes : les quatre membres et le tronc (comprenant la tête). La rotation du tronc étant liée à la rotation ω_1 du personnage, seuls les quatre membres peuvent être adaptés pour corriger la valeur du moment cinétique. Lors de l'adaptation du groupe (membre) K, les autres segments du squelette ne sont pas modifiés. Le moment cinétique est alors donné par l'équation :

$$L = \sum_{i \notin K} I_i \omega_i + \sum_{i \in K} I_i \omega_i + \sum_i m_i G G_i \times \dot{GG_i} \qquad (10)$$

Kulpa et coll. [KMA05] ont montré qu'il était possible de retrouver directement la configuration du groupe K à partir de la position de son centre de masse G_K. La configuration du groupe K n'ayant pas d'influence sur $\sum_{i \notin K} I_i \omega_i$, cette valeur reste donc constante quelque soit l'adaptation réalisée sur K. Cependant, la modification de la position de G_K a une influence sur la position du centre de masse global G du personnage. Soit :

$$h(K) = \sum_{i \in K} I_i \omega_i + \sum_i m_i G G_i \times \dot{GG_i} \qquad (11)$$

$h(K)$ doit donc être évaluée pour chaque modification du membre K.

Si l'on considère qu'un groupe est composé de 2 segments, il y a une unique solution liant la configuration des segments du membre à la position de son centre de masse (figure 1). Ici, la configuration d'un membre est donnée par un vecteur représentant l'extension du membre (l_K sur la figure 1) et par un angle α_K permettant de positionner l'articulation intermédiaire (coude ou genou). Dans le but de préserver le style du mouvement d'origine, nous proposons de préserver l'angle α_K. L'orientation des deux segments du membre est donc exprimée comme une fonction de l_K et α_K. h_K peut ainsi être exprimée directement en fonction de G_K :

$$L_0 - \sum_{i \notin K} I_i \omega_i = h_K(G_K) \qquad (12)$$

Comme dans la section précédente, h_K peut être linéarisée localement en utilisant la matrice Jacobienne J_K :

$$\Delta G_K = J_K^{-1} \left(L - \sum_{i \notin K} i_i \omega_i - \widehat{h_K(G_k)} \right) \qquad (13)$$

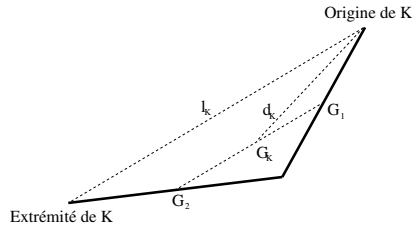


Figure 1: *Membre représenté par une ligne joignant les deux extrémités. La position du centre de masse G_K est liée à la longueur l_K de la ligne.*

où ΔG_K est le déplacement ajouté au centre de masse $\widehat{G_K}$ de la posture d'origine dans le but de conserver la valeur initiale du moment cinétique. Dans certains cas, ΔG_K peut être trop important et, par conséquent, G_K inatteignable. Dans ce cas, la distance d de la figure 1 est plus grande que la valeur maximale d_{max} (membre tendu). Pour résoudre ce problème, G_K est placé sur la ligne joignant l'origine du groupe (l'épaule ou la hanche) à la position $G_K + \Delta G_K$, avec une distance maximale autorisée de d_{max}. Pour éviter d'obtenir des discontinuités dans le mouvement, G_K est filtré avec un filtre passe-bas récursif.

Si l'adaptation d'un groupe n'est pas suffisante pour atteindre la valeur L_0, un autre groupe est utilisé. Ils sont adaptés séquentiellement jusqu'à atteindre la valeur L_0. Toutefois, lorsque l'écart entre le moment cinétique calculé et L_0 est trop important, l'utilisation des quatre membres peut ne pas être suffisante. Dans ce cas, nous utilisons la méthode de la section 4 : la vitesse de rotation globale du personnage est ajustée pour corriger la valeur du moment cinétique.

6. Résultats

Dans le but d'évaluer notre méthode, nous avons demandé à un sujet de réaliser différents mouvements aériens. Ceux-ci ont alors été traités et stockés dans une représentation indépendante de la morphologie, comme suggéré dans [KMA05]. Plusieurs personnages de différentes tailles et masses ont été créés avec le logiciel Avatar Studio (produit de Canal NuMedia). Les paramètres anthropomorphiques des personnages ont été calculés grâce à [Lev96].

Une fois le mouvement mis à l'échelle du personnage, l'accélération du centre de masse n'est plus égale à la gravité et le moment cinétique n'est plus constant (lors des phases aériennes). La figure 2 présente un personnage jouant un salto sur le côté ainsi que les valeurs du moment cinétique corrigé et non corrigé. Sans correction (courbes en traits pleins) on peut remarquer que le moment cinétique n'est pas constant. Après correction, nous obtenons une valeur quasiment constante (courbes pointillées). La séquence obtenue est affichée au bas de la figure. Dans cet exemple, seule la méthode corrigeant la vitesse de rotation globale du person-

nage a été utilisée (section 4). Les temps de calculs sont de l'ordre de 2ms par image pour l'adaptation de la vitesse de rotation. Cela revient à pouvoir simuler environ 16 personnages à 30Hz sans visualisation. Ces résultats ont été obtenus sur un PentiumD 3.4GHz équipé de 2Go de RAM et d'une carte graphique Quadro FX 3450.

La valeur initiale du moment cinétique dépendant de la morphologie du personnage (elle peut varier du simple au double en fonction de la morphologie), il est important de prendre en compte cette valeur lorsque l'on applique un mouvement à un personnage. Dans ce sens, la partie gauche de la figure 3 montre deux personnages identiques jouant le même mouvement. La seule différence réside dans le fait que le personnage noir possède 50% de masse en plus sur les membres inférieurs, entraînant une inertie plus importante et, par conséquent, une vitesse de rotation plus faible. Cette figure démontrent clairement que les changements de la morphologie utilisée se répercutent sur le réalisme dynamique du mouvement obtenu.

Un challenge important dans l'étude et la simulation de mouvements humains est d'être capable de calculer les conséquences d'une modification appliquée au mouvement. Dans le cas des mouvements gymniques, les bras sont utilisés pour initier des vrilles au cours de la phase aérienne. Dans la figure 4, nous ajoutons un mouvement supplémentaire au bras pendant que le personnage réalise le salto arrière, dans le but de vérifier que notre méthode permet de retrouver un comportement naturel. Nous pouvons voir que le personnage noir est orienté différemment que le personnage gris à l'atterrissage (les deux personnages corrigeant leur vitesse de rotation). Le personnage noir a ainsi initié un début de vrille au cours de la phase aérienne, vrille qui n'était pas présente dans le mouvement initial. Nous rappelons que, dans cette démo, seule la phase aérienne est adaptée. Aucun changement n'a été apporté ni à la vitesse initiale ni au moment cinétique initial.

La seconde stratégie consiste à compenser l'action d'un membre en utilisant un ou plusieurs autres membres. La partie droite de la figure 3 montre deux personnages jouant le même mouvement. Le personnage noir joue un mouvement supplémentaire du bras gauche, ce qui introduit une variation du moment cinétique, compensée par l'adaptation du bras droit.

Ces résultats montrent que de telles approches permettent de contrôler et de corriger les phases aériennes des mouvements. Cependant, avec la seconde méthode, de fortes discontinuités apparaissent lorsque l'on travaille sur des mouvements bruités, ou lorsque les membres adaptés sont déjà tendus (il n'est alors pas possible d'adapter la position du centre de masse du membre). Ces problèmes sont liés à une évaluation erronée de la matrice jacobienne.

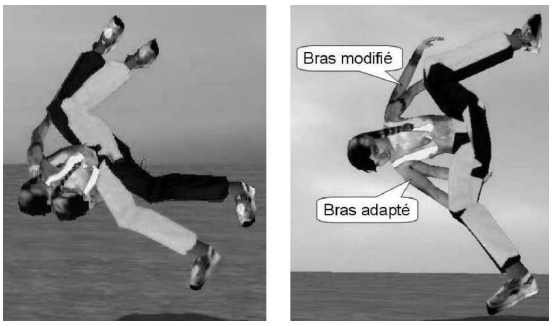


Figure 3: *A gauche : deux personnages ayant des masses identiques pour le haut du corps, mais le personnage noir possède 50% de masse en plus sur la partie inférieure. A droite : le personnage noir joue un mouvement supplémentaire du bras gauche, et contrôle son moment cinétique en adaptant la position de son bras gauche.*

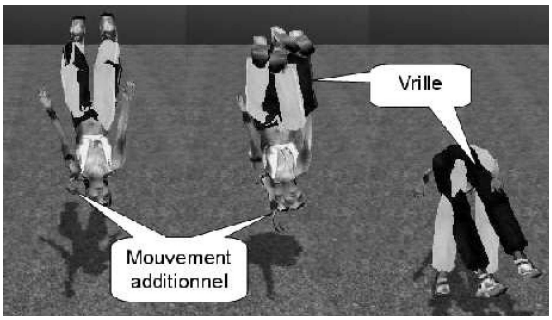


Figure 4: *Le personnage noir joue un mouvement supplémentaire du bras gauche, entraînant la création d'une vrille no nprésente dans le mouvement d'origine.*

7. Discussions et perspectives

Cet article a présenté une méthode permettant d'adapter les mouvements aériens dans le but d'obtenir une séquence dynamiquement correcte. De plus, elle peut être utilisée pour adapter ces mouvements à des personnages différents du sujet d'origine et permet de retrouver des comportements réalistes avec des temps de calculs très faibles. En plus d'être rapide, cette méthode est tout à fait désignée pour l'utilisation en environnement interactif du fait qu'elle ne prend en aucun cas d'informations sur le futur. Elle se comporte donc comme un filtre dynamique, prédisant la posture dynamiquement correcte, mais d'une manière plus efficace. Contrairement aux filtres dynamiques, notre méthode ne nécessite pas de jouer sur différents paramètres. Il suffit de fournir les informations sur la masse des segments du personnage.

La principale limitation de ce type de méthode interactive est liée au contrôle du mouvement du personnage. Comment déterminer le moment cinétique et la vitesse ini-

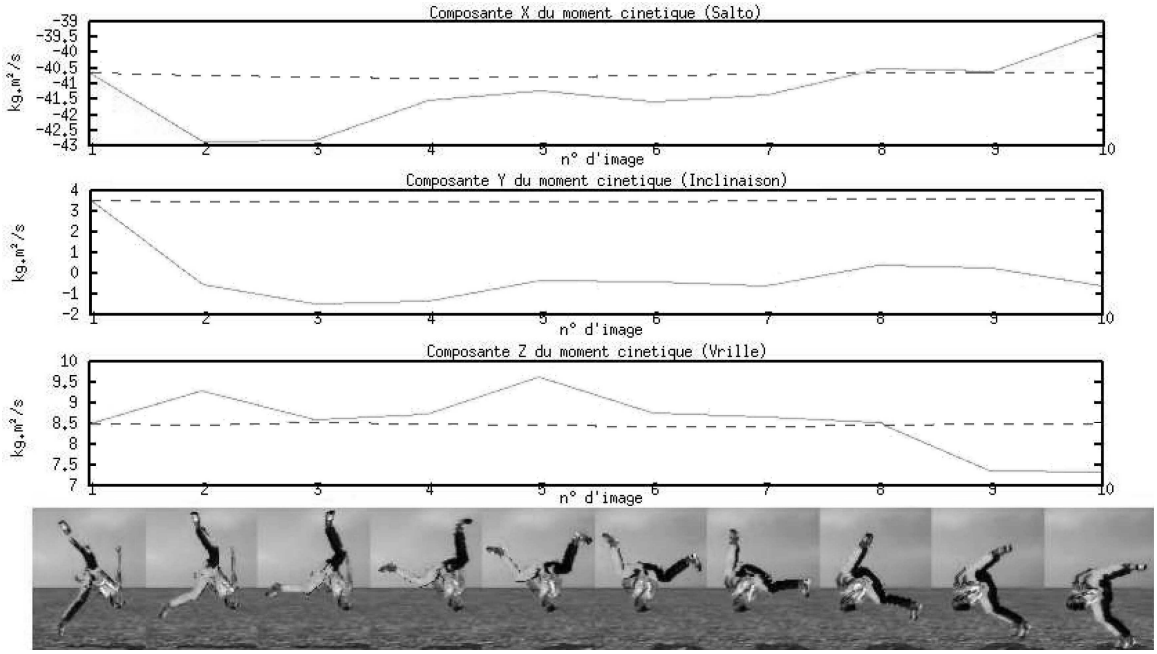


Figure 2: *Composantes du moment cinétique pour un mouvement non corrigé (traits continus, moment cinétique non constant, personnage gris) et corrigé (traits pointillés, moment cinétique constant, personnage noir).*

tiale assurant que les contraintes cinématiques définies par l'utilisateur sont atteintes durant la phase aérienne ? Comme nous n'avons pas d'informations sur le futur et que des modifications extérieures peuvent apparaître à tout moment, il est impossible de prédire quelle seront les postures du personnage au cours du saut. Ce problème peut être en partie résolu en calculant les paramètres dynamiques initiaux sans modification du saut. Si les contraintes sont atteignables, il est alors possible de les vérifier, à condition que d'autres évènements extérieurs n'interviennent pas durant la simulation finale.

La méthode proposée dans cet article est cependant très sensible au bruit haute-fréquence qui entraîne des discontinuités dans les dérivés numériques. Les mouvements aériens en comportant peuvent par conséquent poser problème. Dans ce cas, il pourrait être efficace de subdiviser le pas de correction. Lors de travaux futurs, nous souhaitons adapter cette méthode pour prendre en compte des forces extérieures s'exerçant sur le personnage, telles que des chocs ou des collisions. Par exemple, si un personnage saute et donne un coup de pied ou s'il se fait frapper en plein vol.

La principale application d'une telle méthode est l'animation interactive, telle que la réalité virtuelle ou les jeux vidéos. Par exemple, nous avons commencé à développer une application où un personnage de synthèse joue un mouvement issu d'une base de donnée de mouvements gymniques. Le sujet peut alors contrôler en temps réel le personnage avec un système de capture de mouvements, dans le but d'observer les conséquences de ses gestes sur le mouvement simulé. Ceci pourrait être un pas important dans l'apprentissage et la compréhension de figures gymniques difficiles et dangereuses.

Remerciements

Les auteurs souhaitent remercier le professeur Shigeo Morishima de l'Université de Waseda au Japon, pour les captures de mouvements de gymnastes, ainsi que le Conseil Régional de Bretagne, le Conseil Général et Rennes Métropole pour leur soutien financier.

References

[AFO05] ARIKAN O., FORSYTH D., O'BRIEN J. F.: Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 59–66.

[CK00] CHOI K., KO H.: Online motion retargetting. *The Journal Of Visualisation and Computer Animation 2000 11*, 5 (Dec. 2000), 223–235.

[Coh92] COHEN M.: Interactive spacetime control for animation. *Proceedings of ACM SIGGRAPH '92 .26* (July 1992), 293–302. Chicago, Illinois.

[FP03] FANG A., POLLARD N.: Efficient synthesis of physically valid human motion. *ACM Trans. on Graphics 22*, 3 (2003), 417–426.

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computer & Graphics 25* (2001), 933–953.

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *In Proceedings of Symposium on Interactive 3D Graphics* (1997), pp. 139–148.

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proc. of ACM SIGGRAPH* (July 1998), pp. 33–42.

[Hod95] HODGINS J.: Animating human athletics. *Proceeding of ACM SIGGRAPH'95* (Aug. 1995). Los Angeles, California.

[HP97] HODGINS J., POLLARD N.: Adapting simulated behaviors for new characters. In *Proceedings of ACM SIGGRAPH* (Los Angeles, California, Aug. 1997), Addison Wesley.

[KHL05] KOMURA T., HO E., LAU R.: Animating reactive motion using momentum-based inverse kinematics. *Computer Animation and Virtual Worlds 16* (2005), 213–223.

[KMA05] KULPA R., MULTON F., ARNALDI B.: Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum, Eurographics 2005 special issue 24*, 3 (2005), 343–352.

[Lev96] LEVA P. D.: Adjustments to zatsiorsky-seluyanov s segment inertia parameters. *Journal of Bomechanics 29* (1996), 1223–1230.

[LGC94] LIU Z., GORTLER S., COHEN M.: Hierarchical spacetime control. *Computer Graphics* (July 1994), 35–42. In proceedings of ACM SIGGRAPH'94.

[LHP06] LIU C., HERTZMANN A., POPOVIC Z.: Composition of complex optimal multi-character motions. In *Proceedings of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2006), Eurographics, pp. 215–222.

[LP02] LIU C., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Transaction on Graphics 21*, 3 (July 2002), 408–416.

[LS99] LEE J., SHIN S.: A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of ACM SIGGRAPH 99* (Aug. 1999), 39–48.

[MPS06] MACCANN J., POLLARD N., SRINIVASA S.: Physics-based motion retiming. In *Proceedings of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (Vienna, Austria, 2006), Eurographics, pp. 205–214.

[PBM00] POLLARD N., BEHMARAM-MOSAVAT F.: Force-based motion editing for locomotion tasks. In *Proceedings of IEEE international conference on Robotics and Automation* (San Francisco, CA, April 2000).

[RGBC96] ROSE C., GUENTER B., BODENHEIMER B., COHEN M. F.: Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 147–154.

[SH05] SAFONOVA A., HODGINS J.: Analyzing the physical correctness of interpolated human motion. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM Press, pp. 171–180.

[SHP04] SAFONOVA A., HODGINS J., POLLARD N.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *Proceedings of ACM SIGGRAPH* (2004).

[SKG03] SHIN H., KOVAR L., GLEICHER M.: Physical touch-up of human motions. In *Proceedings of Pacific Graphics* (Alberta, Canada, October 2003).

[TK05] TAK S., KO H.: A physically-based motion retargeting filter. *ACM Transactions on Grpahics 24*, 1 (2005), 98–117.

[TSK02] TAK S., SONG O., KO H.: Spacetime sweeping: a interactive dynamic constraints solver. In *Proceedings of IEEE Computer Animation* (June 2002), pp. 261–270.

[vdPF93] VAN DE PANNE M., FIUME E.: Sensor-actuator networks. In *Proceedings of ACM SIGGRAPH* (Anaheim, California, Aug. 1993), Addison Wesley, pp. 335–342.

[WC91] WANG L.-C. T., CHEN C. C.: A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. On Robotics and Applications 7*, 4 (August 1991), 489–499.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Proceedings of ACM SIGGRAPH* (Atlanta, Georgia, Aug. 1988), Addison Wesley, pp. 159–168.

[Yea93] YEADON M.: The biomechanics of twisting somersaults. part ii. contact twist. *Journal of Sports Sciences 11* (1993), 199–208.

[YN03] YAMANE K., NAKAMURA Y.: Dynamic filters - concept and implementations of online motion generator for human figures. *IEEE transactions on robotics and automation 19*, 3 (2003), 421–432.

[ZH99] ZORDAN V., HODGINS J.: Tracking and modifying upper-body human motion data with dynamic simulation. In *Proc. of EGCAS'99* (Sept. 1999), pp. 13–22.

[ZH02] ZORDAN V., HODGINS J.: Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (San Antonio, Texas, 2002), ACM Press, pp. 89–96.

[ZMCF05] ZORDAN V., MAJKOWSKA A., CHIU B., FAST M.: Dynamic response for motion capture animation. *ACM Trans. Graph. 24*, 3 (2005), 697–701.

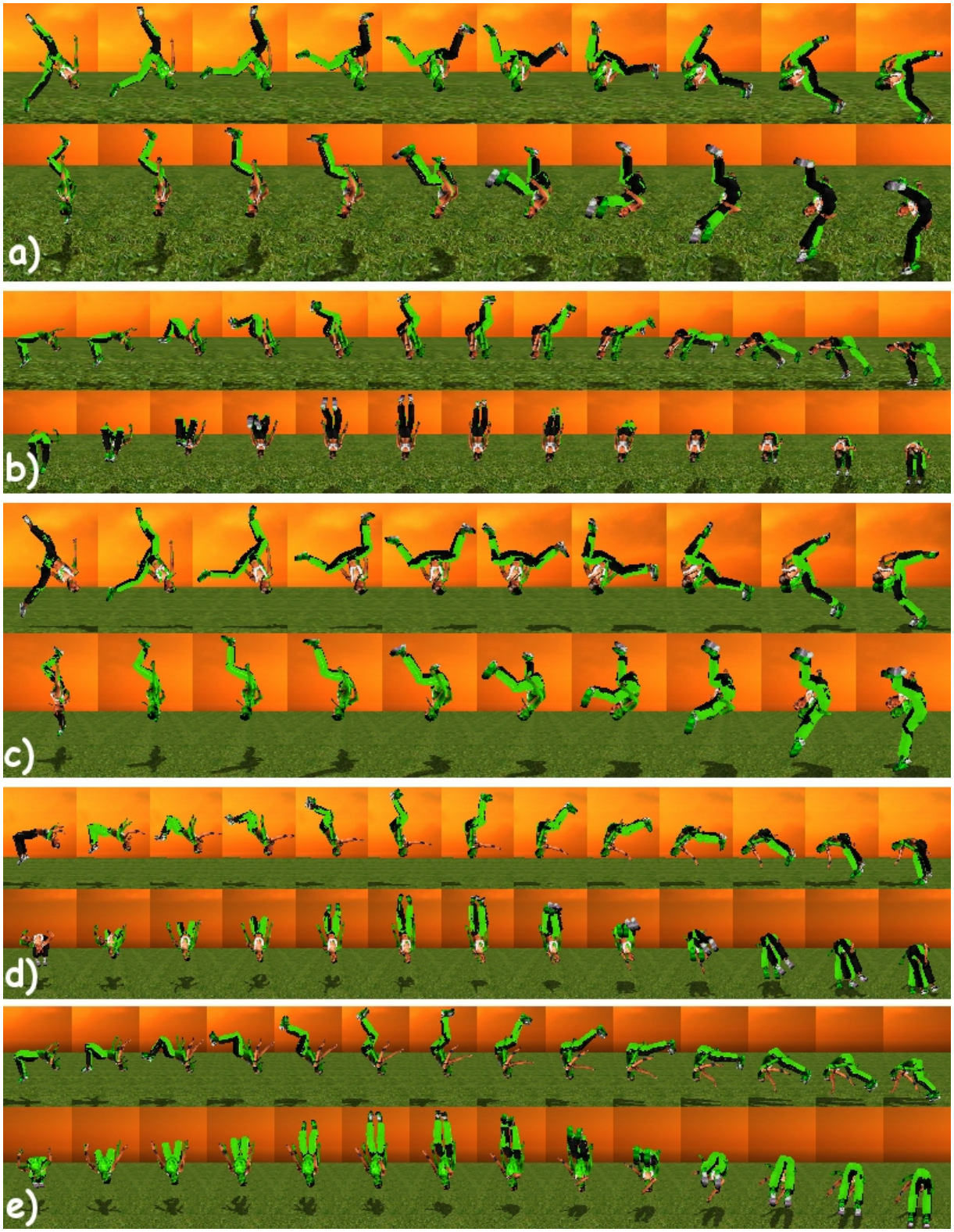


Figure 5: *Aperçu des résultats obtenus sur quelques mouvements aériens avec (personnage texturé) et sans (personnage vert) correction, pour a) un saut sur le côté, b) un salto arrière, e) le personnage vert corrige le mouvement du bras droit en utilisant son bras gauche. c) et d) Les deux personnages corrigent la dynamique. Pour le c), mouvement de salto sur le côté : le personnage texturé a 50% de lasse supplémentaire sur les membres inférieurs et tourne donc moins vite. d) Salto arrière : le personnage texturé joue un mouvement supplémentaire du bras, entraînant la création d'un début de vrille.*

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Modélisation de formes vapoureuses à l'aide d'IFS

Dmitry Sokolov, Christian Gentil et Marc Neveu

Laboratoire LE2I — UMR CNRS 5158
Aile de l'Ingénieur, Faculté des Sciences Mirande, Université de Bourgogne
21000 Dijon, France

Abstract

En synthèse d'images, les tentatives de modélisation de gaz ont commencé vers la fin des années soixante-dix. Depuis, de nombreuses méthodes ont été proposées. Cependant, la majorité d'entre elles sont trop complexes pour les graphistes et artistes peu familiarisés avec les mathématiques et la physique. Dans cet article nous présentons une nouvelle méthode intuitive pour la création d'objets vaporeux tels que des nuages ou des personnages de fumée. Bien que cette méthode soit basée sur la géométrie fractale, le graphiste n'a pas à être effrayé, puisqu'aucune connaissance mathématique n'est nécessaire. Le principe consiste a créer quelques esquisses qui décriront la forme finale de l'objet vaporeux. Les avantages de cette nouvelle méthode sont : la simplicité, le contrôle très précis du résultat, la facilité d'animation des objets.

1. Introduction

La modélisation des formes présentes dans la nature telles que les montages, nuages, feux, fumées,... a été un défi à relever dans le sens où les modèles classiques, initialement introduits pour les sytèmes de CAD, étaient adaptés à la description de formes lisses et régulières. De nombreux travaux ont relevé ce défi et maintenant un ensemble d'outils conséquent existe.

Une première famille de méthodes est présentée par David Ebert dans [EMP*02]. Celles-ci sont basées sur la notion de textures procedurales 3D. Ces textures sont générées à partir de bruits pseudo-aléatoires qui permettent de définir une forme en trois dimension "perturbée" à laquelle, le cas échéant, est ajouté un traitement de rendu spécifique pour simuler la nature du matériau souhaité (nuage, fumée, feu,...). Le principe consiste à construire une "forme support" qui représente la forme générale souhaitée. A cette forme initiale est ajouté un bruit structuré de type: bruit de Perlin, fractale, turbulence,... permettant de déformer ou perturber la "forme support" et de la rendre irrégulière. Cette perturbation peut être utilisée à différents niveaux de rendu, bump mapping, displacement mapping, ou directement comme texture de transparence, en fonction de l'effet souhaité. Suivant le bruit utilisé, un certain nombre de paramètres permettent de contrôler les caractéristiques des perturbations. L'animation de ces formes peut se faire à deux niveaux. Le premier consiste à déformer la forme support comme on pourrait le faire classiquement en CAD. Le deuxième consiste à intervenir sur le bruit. Mais dans ce cas, la génération du bruit faisant appel à un processus pseudo-aléaoire, le contrôle est plus délicat notamment si l'on souhaite avoir des propriétés de continuité au niveau de l'animation. Une solution consiste à générer le bruit avec une dimension suplémentaire qui correspond à la variation de la texture dans le temps. La difficulté est alors de contrôler cette variation en fonction du temps, car étant construite à partir du même type bruit elle est par nature difficiles à contrôler.

Ce premier type de solutions cherche à donner un effet final ressemblant au phénomène que l'on souhaite représenter. Une autre approche consiste à utiliser un modéle décrivant le phénomène naturel donnant naissance aux formes. Ces modèles basés sur la physique se présentent généralement sous forme d'un système dynamique, c'est-à-dire un ensemble d'équations qui décrit l'évolution de la forme en fonction de son état. Ces équations sont utilisées pour réaliser une simulation du phénomène de création et d'évolution des formes. A titre d'exemples, nous pouvons mentionner les travaux de Harris et al. [HBSL03] et de Miyazaki et al. [MYND01]. Le contrôle de l'animation est alors géré par les paramètres du système de simulation. Le rendu est très réaliste mais le contrôle précis est plus difficile. Afin d'accroître ce contrôle tout en conservant un réalisme poussé, certains auteurs

combinent les deux méthodes [SDE05]. L'utilisateur dispose d'une interface lui permettant de définir une forme support qui sera utilisée pour contrôler la forme mais à laquelle sera appliquée une méthode de simulation.

Nous avons choisi de développer une nouvelle alternative de production de formes nuageuses en mettant en avant le contrôle des formes par rapport au rendu réaliste. Nous souhaitons proposer un outil très intuitif pour les graphistes et artistes non familiarisés avec les langages mathématiques et physiques, leur permettant de construire une forme précise, reconnaissable et proche d'un croquis initial.

Afin de répondre à ces contraintes, nous nous sommes orientés vers un modèle basé sur la géométrie fractale et offrant un contrôle précis des formes. Il s'agit du modèle IFS (Iterated Function System) introduit par Huchinson et développé par Barnsley. Dans cet article, nous proposons une généralisation du modèle IFS avec ensembles de condensation. Cette généralisation, répond aux critères classiques des IFS et permet d'hériter des propriétés établies de ce modèle : croissance, continuité,... qui garantissent la contrôlabilité des formes modélisées. Nous verrons que cette généralisation est particulièrement adaptée à la génération de formes nuageuses, avec un contrôle simple et intuitif ne nécessitant aucune connaissance mathématique du modèle. La figure 1 montre un exemple de "vaisseau fantôme" créé à partir de cette méthode.

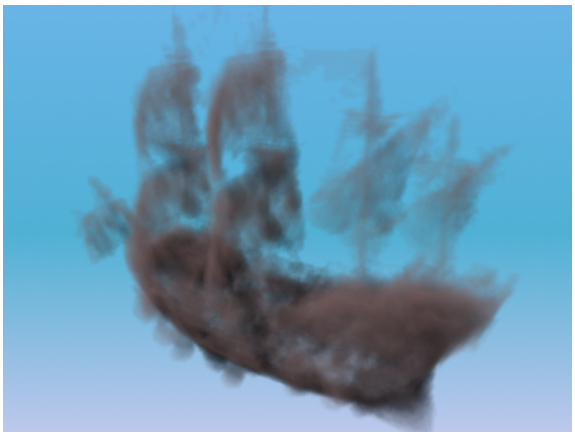


Figure 1: *Un "vaisseau fantôme" peut être facilement créé à l'aide de notre nouvel outil de modélisation.*

Cet article est organisé de la façon suivante: Le paragraphe 2 rappelle les notions de base de la géométrie fractale. Puis, les paragraphes 3 et 4 présentent une généralisation du modèle IFS avec ensembles de condensation. Une série d'exemples est ensuite montrée dans le paragraphe 5. Finalement, le paragraphe 6 conclut cet article et présente les perspectives de ces travaux.

2. Définitions

2.1. Définitions générales

Un *iterated function system* ou *IFS* consiste en un ensemble fini de transformations $f_i : \mathbb{X} \to \mathbb{X}$ pour $i = 1,2,\ldots,N$, où $N \geq 1$ est un entier, $(\mathbb{X},d)$ est un espace metrique complet. On peut le noter $\mathcal{F} = \{\mathbb{X}; f_1, f_2, \ldots f_N\}$.

Une transformation $f_i : \mathbb{X} \to \mathbb{X}$ est *strictement contractante* si et seulement si il existe un réel $0 \leq s_i < 1$ tel que $d(f_i(x), f_i(y)) \leq s_i d(x,y)$ pour tout $x,y \in X$. s_i est appelé constante de contraction de f_i. Si un IFS est composé de transformations strictement contractantes, il est appelé *IFS hyperbolique* et le réel $s = \max\{s_1, s_2 \ldots s_N\}$ est la constante de contraction de l'IFS. Soit $(\mathbb{X},d)$ un espace métrique complet. Alors $\mathbb{H}(\mathbb{X})$ représente l'espace dont les points sont les sous-ensembles compacts de $\mathbb{X}$, autres que l'ensemble vide. Etant donné un IFS hyperbolique, définissons une transformation contractante (au sens de la métrique de Hausdorff $d_{\mathbb{H}}$) $\mathcal{F} : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ comme suit:

$$\mathcal{F}(B) = f_1(B) \cup f_2(B) \cup \cdots \cup f_N(B) \text{ pour tout } B \in \mathbb{H}(\mathbb{X}).$$

Notons que l'on utilise $\mathcal{F}$ pour représenter aussi bien un IFS ou sa transformation correspondante lorsque le contexte n'est pas ambigü. Dans la littérature la transformation $\mathcal{F}$ est aussi appelée l'opérateur de Hutchinson .

Il est notoire [Hut81] qu'il existe un point fixe unique $A_{\mathcal{F}} \in \mathbb{H}(\mathbb{X})$ tel que $A_{\mathcal{F}} = \mathcal{F}(A_{\mathcal{F}})$. En outre, le théorème des applications contractantes [Bar88] stipule que pour tout sous-ensemble compact non vide $B_0 \in \mathbb{H}(\mathbb{X})$ la séquence définie récursivement par $B_{k+1} = \mathcal{F}(B_k)$, converge vers $A_{\mathcal{F}}$ pour la métrique de Hausdorff $k \to \infty$. L'ensemble résultant ne dépend pas du compact non-vide initial B_0, et pour cette raison $A_{\mathcal{F}}$ est appelé *attracteur de l'IFS (hyperbolique)*.

2.2. L'algorithme déterministe

Il est possible de calculer des attracteurs de manière directe au moyen du théorème précédent. Ce procédé est appelé l'algorithme déterministe. Considérons un exemple dans lequel l' IFS consiste en trois transformations affines sur $\mathbb{C}$ (la figure 2 illustre cet exemple):

$$f_1 = \frac{1}{2}z, \quad f_2 = \frac{1}{2}z + \frac{1}{2}, \quad f_3 = \frac{1}{2}z + \frac{1}{2}i.$$

la première étape consiste à choisir arbitrairement un sous-ensemble compact non vide de $\mathbb{C}$. Alors le processus itératif applique chaque transformation f_1, f_2 et f_3 à l'ensemble courant, et ensuite prend l'union des ensembles résultants. Puis f_1, f_2 et f_3 sont appliqués à l'union et ainsi de suite. La limite de ce processus donne l'attracteur de l'IFS, comme indiqué ci-dessus.

2.3. L'algorithme itératif aléatoire

Une autre approche pour générer des attracteurs d'IFS est l'algorithme itératif aléatoire [BD85]. Un *IFS avec proba-*

Figure 2: *Une illustration de l'algorithme déterministe. Partant d'un sous-ensemble compact quelconque (la fleur par exemple), la séquence converge vers le triangle de Sierpinski.*

bilités consiste en un IFS et une distribution de probabilités $p_1, p_2, \ldots, p_N$, ensemble de nombres réels positifs tels que $p_1 + p_2 + \cdots + p_N = 1$. Un IFS avec probabilités peut être noté $\{\mathbb{X}; f_1, f_2, \ldots, f_N; p_1, p_2, \ldots, p_N\}$. la probabilité p_i est associée avec la fonction de transformation f_i pour tout $i \in \{1, 2, \ldots, N\}$.

Si on utilise le plan $\mathbb{C}$ comme espace d'itération $\mathbb{X}$, alors l'algorithme itératif aléatoire (aussi appelé "jeu de chaos") peut être exprimé en pseudo-code par:

1 **Données:** Un IFS avec probabilités $\mathcal{F}$
2 **Résultats:** Une approximation (une image) de l'ensemble attracteur $A_{\mathcal{F}}$
3 sélectionner un point aléatoirement $z_0 = (x_0, y_0)$ du plan $\mathbb{C}$
4 itérer {
5 choisir une transformation f_i selon les probabilités préassignées $\{p_1, p_2, \ldots, p_N\}$.
6 $z_{k+1} = f_i(z_k)$
7 if ($k > 100$) tracer le pixel de coordonnées z_{k+1}
8 }

Ce schéma fonctionne parce que l'IFS est contractant: si z_k est proche de l'attracteur $A_{\mathcal{F}}$, alors $f_i(z_k)$ est encore plus proche. Bien que l'algorithme démarre avec un point aléatoire, la distance entre l'ensemble solution $A_{\mathcal{F}}$ et le point "fluctuant" decroît exponentiellement. Dans cet exemple on a pris 100 itérations avant de tracer un point. En général, la distance entre $A_{\mathcal{F}}$ et z_{101} est inférieure à la taille d'un pixel. En outre, le théorème ergodique d'Elton [Bar88, p. 370] stipule que, avec la probabilité 1, l'ensemble $\{z_k\}$ est dense dans $A_{\mathcal{F}}$. Il n'y a pas de nombre d'itérations fixé pour l'algorithme. Puisque le jeu de chaos opère par échantillonnage stochastique, plus on fait d'itérations plus le résultat est proche de la solution exacte. S'il n'y a pas de probabilités prédéfinies, on peut choisir les transformations avec une fréquence égale dans la ligne 5 du jeu de chaos.

Un exemple d'IFS avec probabilités est $\{\mathbb{C}; f_i; p_i; i = 1,2,3,4\}$, où $p_i = 0.25$ et

$$f_1(z) = \frac{1}{2}z, \quad f_2(z) = \frac{1}{2}z + \frac{1}{2}$$
$$f_3(z) = \frac{1}{2}z + \frac{1}{2}i, \quad f_4(z) = \frac{1}{2}z + \frac{1}{2} + \frac{1}{2}i.$$

Voici comment le jeu de chaos procède dans le cas présent. Un point initial $z_0 \in \mathbb{C}$ est choisi. Une des transformations est sélectionnée "au hasard" dans l'ensemble $\{f_1, f_2, f_3, f_4\}$. La transformation sélectionnée est appliquée à z_0 pour produire un nouveau point $z_1 \in \mathbb{C}$. De nouveau une transformation est sélectionnée, de la même façon, indépendamment du choix précédent, et appliquée à z_1 pour produire un nouveau point z_2. Le processus est répété un certain nombre de fois donnant ainsi une séquence finie de points $\{z_k : k = 1, 2, \ldots, \textit{nombre d'iterations}\}$. Partant d'une certaine étape, la fin de la séquence est située dans le carré unité ■ de sommets (0, 0), (1, 0), (1, 1) et (0, 1). Si le nombre d'itérations est suffisamment grand, le résultat sera l'image du carré unité rempli.

Dans cet exemple le point "fluctuant" visite tous les points de ■ avec une égale probabilité. Que se passe-t-il si nous changeons les probabilités? Considérons l'IFS ci-dessus avec un ensemble de probabilités différent $\{0.30, 0.20, 0.25, 0.25\}$. Changeons légèrement les règles du jeu: au lieu d'afficher simplement un pixel z_k, comptons combien de fois chaque pixel a été visité par le point "fluctuant". Quand l'algorithme s'achève, on trace une image en niveaux de gris dans laquelle les pixels clairs figurent aux places les plus visitées et les plus sombres aux places les moins visitées.

Il est surprenant de voir que l'image change complètement! Se référer à la figure 3, où l'image de gauche montre le résultat avec une égale probabilité, alors que celle de droite montre le résultat avec le nouvel ensemble de probabilités.

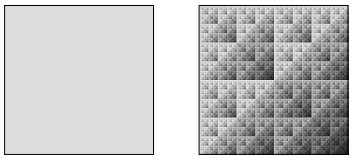


Figure 3: *Attracteurs de mesure : différents distributions de probabilités conduisent à différentes distributions de "masse" dans attracteur. L'image de gauche est calculée avec une égale probabilité* $\{0.25, 0.25, 0.25, 0.25\}$*, celle de droite est obtenue avec l'ensemble* $\{0.30, 0.20, 0.25, 0.25\}$.

Ces images traduisent une merveilleuse idée. Elles suggèrent qu'associée avec un IFS avec probabilités il existe une unique "densité" sur l'attracteur de l'IFS. Barnsley [Bar06, pp. 122–129] a développé l'idée de densités en utilisant le formalisme du théorème de la mesure. Les Mesures peuvent être utilisées pour décrire une distribution de masse complexe dans des espaces métriques. Etant donné un IFS $\mathcal{F} = \{\mathbb{X}; f_i; p_i; i = 1, 2, \ldots, N\}$, Barnsley introduit un autre IFS $\{\mathbb{P}(\mathbb{X}); f_i; p_i; i = 1, 2, \ldots, N\}$, où $(\mathbb{P}(\mathbb{X}), d_{\mathbb{P}})$ est l'espace des mesures normalisées de Borel sur $\mathbb{X}$ avec la métrique de Monge-Kantorovich. Il montre qu'une transformation f contractante sur $\mathbb{X}$ hérite de cette propriété sur $\mathbb{P}(\mathbb{X})$. Dès lors, la nouvelle transformation $\mathcal{F} : \mathbb{P}(\mathbb{X}) \to \mathbb{P}(\mathbb{X})$ a un unique point fixe appelé *mesure fractale* ou *attracteur de mesure*.

En revenant à notre exemple, les deux IFS ont le même *ensemble* attracteur (le carré unité ■), mais différents attracteurs de *mesure* .

3. Espace de sous-ensembles compacts

Bien que la théorie des IFS traite d'espaces métriques complets arbitraires, presque toutes les implémentations construisent des fractales dans un plan bi-dimensionel avec des contractions affines. Très peu d'approches utilisent l'espace tridimensionnel et/ou des transformations non-linéaires. Par exemple, se référer aux travaux de Scott Draves [Dra05, Dra06].

3.1. IFS avec condensation

Un pas a été franchi avec Barnsley en 1988 quand il a introduit la notion d'IFS avec condensation (constante) : soit $(\mathbb{X}, d)$ un espace métrique et soit $C \in \mathbb{H}(\mathbb{X})$. Alors une transformation constante $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ définie par $f_0(B) = C$ pour tout $B \in \mathbb{H}(\mathbb{X})$ est appelée une *transformation de condensation* et C est appelé *l'ensemble de condensation associé*. Une transformation de condensation $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ est une application contractante sur l'espace métrique $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$ avec un facteur de contraction égal à zéro. Donc elle possède un point fixe unique, qui est l'ensemble de condensation.

A partir de là, Barnsley définit un nouvel IFS: soit $\{\mathbb{X}; f_1, f_2, \ldots, f_n\}$ un IFS hyperbolique de constante de contraction $0 \leq s < 1$. Soit $f_0 : \mathbb{H}(\mathbb{X}) \to \mathbb{H}(\mathbb{X})$ une transformation de condensation. Alors $\{\mathbb{X}; f_0, f_1, \ldots, f_n\}$ est appelé *IFS hyperbolique avec condensation*, de constante de contraction s.

Considérons un exemple, où un IFS consiste en une contraction $f_1(z) = \frac{1}{2}z$. Manifestement l'attracteur est un simple point, l'origine. Mais si nous ajoutons une transformation de condensation f_0 qui transforme tout sous-ensemble compact en un "pin", alors l'attracteur devient une série de "pins". Voir l'illustration figure 4 .

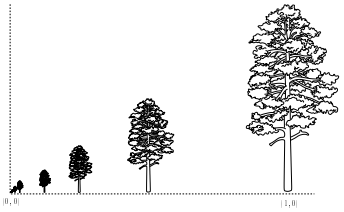


Figure 4: *une série géométrique de pins, l'attracteur d'un IFS avec condensation.*

3.2. Implémentation généralisée

Nous souhaitons enrichir la notion d'IFS avec condensation. L'implémentation que nous réalisons nécessite des transformations de l'espace des sous-ensembles compacts $\mathbb{H}(\mathbb{X})$. En effet, un IFS est composé de contractions sur $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$. Une contraction f sur $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$ est une application avec un point fixe unique. Cela signifie que pout tout sous-ensemble compact non-vide B_0 la séquence $\{B_k\}$, définie récursivement par $B_{k+1} = f(B_k)$ converge vers le point fixe (selon la métrique de Hausdorff). Dans son approche Barnsley choisit un point fixe C (le "pin" dans la figure 4). Dès lors, il est clair que l'application constante qui correspond est une contraction.

La question qui se pose est la suivante: est-il possible de trouver des applications contractantes sur $\mathbb{H}(\mathbb{X})$ autres que des applications constantes? Oui bien sûr. Voir la figure 5 par exemple. Etant donné un ensemble C, il suffit de construire une application f telle que *tout* sous-ensemble compact non-vide soit transformé en l'ensemble C par application récursive de f.

Plus rigoureusement, le théorème de Meyer réciproque du théorème de Banach [Mey67] stipule que si pour tout $B_0 \in \mathbb{H}(\mathbb{X})$ la séquence $\{B_k\}$, définie récursivement par $B_{k+1} = f(B_k)$, converge vers C uniformément sur un voisinage de C au sens de la métrique de Hausdorff $d_{\mathbb{H}}$, alors il y a une métrique topologiquement équivalente $d'_{\mathbb{H}}$ sur $\mathbb{H}(\mathbb{X})$ qui rend f strictement contractante (en fait l'algorithme fonctionne sous la condition faible qui est que le système entier est contractant en moyenne).

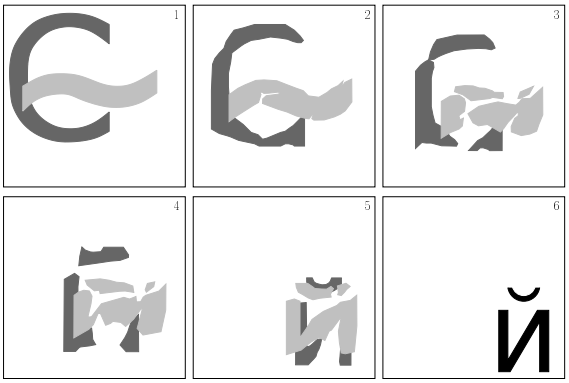


Figure 5: *Une application contractante sur* $(\mathbb{H}(\mathbb{X}), d_{\mathbb{H}})$. *Le point fixe (choisi par l'utilisateur) est en noir sur l'image finale. Si tout sous-ensemble compact non-vide est un morphisme de l'ensemble donné par application récursive de la transformation alors la transformation est contractante. Deux exemples en gris foncé et gris clair. L' image numéro 4 montre les ensembles médians correspondants.*

Pratiquement on peut imaginer l'espace $\mathbb{X}$ comme une feuille de papier blanc, et l'espace $\mathbb{H}(\mathbb{X})$ comme l'ensemble de toutes les impressions noires possibles sur la feuille. Pour simplifier la présentation, nous parlerons d'images bi-dimensionnelles, bien que toutes les techniques puissent être appliquées à des espaces arbitraires.

Dans cette section nous construirons un exemple d'une

telle transformation. Le problème de définir des applications contractantes dans $\mathbb{H}(\mathbb{X})$ est intimement lié au problème de l'interpolation d'images (morphing d'images binaires). L'interpolation d'images est un terme général pour désigner un ensemble de techniques utilisées en Infographie pour générer des images intermédiaires entre deux images données. Par exemple, Iwanowski et Serra [IS00] proposent une méthode de morphing consistant en trois étapes (la figure 6 illustre le procédé):

1. D'abord, pour deux images binaires données on détecte les boîtes englobantes pour superposer les images.
2. Puis les auteurs calculent l'image médiane morphologique.
3. Enfin, ils placent l'image médiane à sa position finale.

L'étape clé est le calcul de l'ensemble médian. Les auteurs définissent l'ensemble médian $M(A,B)$ entre deux ensembles A et B comme suit:

$$M(A,B) = \{x \in \mathbb{X} : d(x, A \cap B) < d(x, \overline{A \cup B})\},$$

où d est la pseudo-distance euclidienne d'un point à un ensemble. En d'autres termes, l'ensemble médian est formé de points plus proches de l'intersection entre A et B que du complément de leur union. Ayant déterminé une méthode de calcul d'ensemble médian, il est facile de calculer une séquence nécessaire d'ensemble de transformations:

- Première itération: $M_{1/2} = M(A,B)$.
- Seconde itération:
  1. $M_{1/4} = M(A, M_{1/2})$
  2. $M_{3/4} = M(M_{1/2}, B)$
- Troisième itération:
  1. $M_{1/8} = M(A, M_{1/4})$
  2. $M_{3/8} = M(M_{1/4}, M_{1/2})$
  3. $M_{5/8} = M(M_{1/2}, M_{3/4})$
  4. $M_{7/8} = M(M_{3/4}, B)$
- et ainsi de suite.

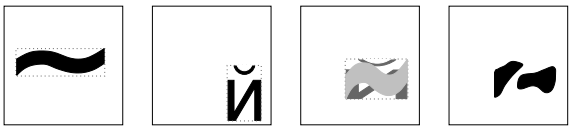


Figure 6: *La première image représente l'image source, la seconde l'image de destination. L'idée est de superposer les images d'entrée sur la position centrale (troisième image) et ensuite de calculer l'image médiane (celle de droite).*

Malheureusement, la méthode présente beaucoup de restrictions. D'abord, l'union $A \cup B$ ne doit pas être égale à $\mathbb{X}$. De plus, l'intersection $A \cap B$ ne doit pas être vide. Par ailleurs, l'intersection entre toutes les composantes connexes ne doit pas être vide.

Pour éviter ces limitations, nous avons développé une méthode simple de morphing d'images binaires. Ainsi, nous avons deux images A et B. Définissons une image médiane *gauche* $M_l(A,B)$ construite selon les règles suivantes:

1 **Données:** Deux images binaires A et B
2 **Résultat:** Image médiane gauche $M_l(A,B)$
3 Effacer la feuille $M_l(A,B)$
4 Pour tout point noir p_A de A {
5 trouver le point noir correspondant le plus proche p_B dans l'image B
6 tracer un point dans l'image $M_l(A,B)$ au centre du segment (p_A, p_B)
7 }

Ainsi, l'image médiane gauche $M_l(A,B)$ est l'image A déformée par l'influence de B. L'image médiane droite $M_r(A,B)$ est définie de la même façon, avec l'image B déformée. Enfin l'image médiane finale est définie comme union des images droite et gauche: $M(A,B) = M_l(A,B) \cup M_r(A,B)$. La figure 7 montre une illustration. Pour des raisons pratiques tous les ensembles sont tracés sur la même feuille de papier, mais en fait la figure représente cinq images superposées. L'image A est une virgule tracée à gauche d'une feuille blanche, l'image B représente le caractère à droite sur une autre feuille etc.

Figure 7: *un exemple de séquence de morphing. Les ensembles médians gauche et droit sont indiqués par les couleurs gris foncé et gris clair.*

Ayant fixé un ensemble $C \in \mathbb{H}(\mathbb{X})$, l'application $M(\cdot, C)$ est contractante au sens de la métrique de Hausdorff:

$$\lim_{q \to 1} d_{\mathbb{H}}(M_q(X,C), C) = 0 \ \ \forall X \in \mathbb{H}(\mathbb{X}).$$

Nous appellerons l'ensemble C *ensemble de condensation* pour une application contractante dans $\mathbb{H}(\mathbb{X})$.

En résumé: nous travaillons avec des IFS de type $\{\mathbb{H}(\mathbb{X}); f_1, f_2 \ldots, f_n\}$. L'opérateur de Hutchinson $\mathcal{F}$ est défini de $\mathbb{H}(\mathbb{H}(\mathbb{X}))$ sur $\mathbb{H}(\mathbb{H}(\mathbb{X}))$. Intuitivement le point fixe $A_{\mathcal{F}}$ est un ensemble de feuilles de papier sur lesquelles on a dessiné des figures. Toutes les transformations f_i ont des sous-ensembles compacts comme points fixes (c'est à dire que le point fixe de la transformation f_i est une feuille de papier peint). L'attracteur $A_{\mathcal{F}}$ contient tous les points fixes. Donc, si $f_i := M(\cdot, C_i)$, alors il est possible de contrôler la forme de l'attracteur en choisissant les ensembles de condensation C_i de façon appropriée.

Notons que toutes les applications contractantes dans $\mathbb{X}$ sont des applications contractantes dans $\mathbb{H}(\mathbb{X})$. Par exemple, une transformation affine bi-dimensionnelle contractante est

une application contractante dans $\mathbb{H}(\mathbb{X})$, où l'ensemble de condensation correspondant est un point. Donc, cette nouvelle implémentation englobe l'ancienne, héritant de toutes ses propriétés dont la contraction.

4. Nouveau jeu de chaos

Considérons un exemple d'IFS composé de deux transformations: $\mathcal{F} = \{\blacksquare; f_1, f_2\}$, où $\blacksquare$ représente la carré bi-unitaire de sommets (-1, -1), (1, -1), (1, 1) et (-1, 1). La première transformation f_1 est définie à partir de l'ensemble de condensation illustré par l'image de gauche de la figure 8. La seconde transformation est non-linéaire et telle que $f_2(r,\theta) = \left(\frac{r}{\sqrt{2}}, 2\theta\right)$. En d'autres termes, elle transfome le carré bi-unitaire en l'étirant, de telle sorte que pour un point de coordonnées polaires (r,θ) l' angle θ est doublé. Puis, le carré transformé est réduit de manière à être ajusté à la taille initiale du carré bi-unitaire $\blacksquare$. On remarque que l'origine reste inchangée. L'image du milieu de la figure 8 présente l'effet de l'application de f_2 sur une grille définie dans le carré $\blacksquare$.

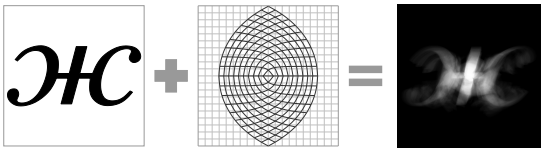


Figure 8: *IFS composé de deux transformations: f_1 dont l'ensemble de condensation correpond à l'image de gauche et $f_2(r,\theta) = \left(\frac{r}{\sqrt{2}}, 2\theta\right)$. L'attracteur de mesure correspondant est rendu par l'image de droite.*

Maintenant, réalisons le rendu de l'attracteur de mesure à l'aide de l'algorithme itératif aléatoire.

1 **Entrées:** Un IFS avec une distribution de probabilités $\mathcal{F} = \left\{\blacksquare;\ f_1, f_2;\ p_1 = \frac{1}{2}, p_2 = \frac{1}{2}\right\}$, une résolution d'image (w,h)
2 **Sortie:** Une approximation (image numérique) de l'attracteur de mesure $\mu_{\mathcal{F}}$
3 dessiner une image binaire aléatoire non-vide I_0 de résolution $w \times h$
4 initialiser une matrice M de taille $w \times h$ à zéro
5 itérer {
6 choisir aléatoirement une transformation f_i en tenant compte des probabilités $\{p_1, p_2\}$
7 $I_{k+1} = f_i(I_k)$
8 si $(k > 100)$ {
9 pour chaque pixel noir (i,j) de I_{k+1}
10 $M(i,j)++$
11 }
12 }
13 Dessiner l'image de niveaux de gris $\mu_{\mathcal{F}}$ correspondant aux valeurs de M

La principale différence avec le jeu du chaos classique et cet algorithme réside dans le fait qu'à chaque itération, il ne faut pas mémoriser uniquement $x_k \in \mathbb{X}$, mais l'image d'un compact non-vide $I_k \in \mathbb{H}(\mathbb{X})$.

5. Autres exemples

5.1. Deux dimensions

Considérons les ensembles A et B de la figure 7. Construisons un IFS $\mathcal{F} = \{\mathbb{X}; f_1, f_2\}$ composé de deux transformations contractantes, obtenues comme exposé au paragraphe 3.2 à partir des ensembles de condensation respectifs A et B.

Ainsi, $f_1 = M(\cdot, A)$, $f_2 = M(\cdot, B)$ et $\mathbb{X}$ représente la feuille de papier rectangulaire. L'attracteur de mesure $\mu_{\mathcal{F}}$ correspondant à l'IFS est présenté figure 9. Dans un tel cas, l'attracteur de mesure montre un procédé de transformation uniforme de A vers B.

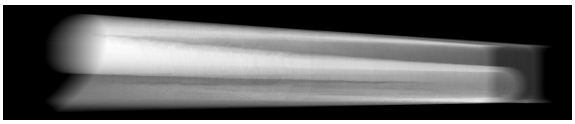


Figure 9: *Image de l'attracteur de mesure d'un IFS composé de deux contractions simples avec ensembles de condensation. Les ensembles de condensation sont présentés par la figure 7.*

Cependant, il existe beaucoup d'autres transformations contractantes possédant les mêmes ensembles de condensation. Par exemple, on peut modifier légèrement la méthode de construction du morphing décrite au paragraphe 3.2. Après avoir détecté les boîtes englobantes des ensembles A et B, on les superpose, puis on leur applique une rotation d'un angle proportionnel à la distance entre A et B, et enfin on calcule l'ensemble médian. L'angle doit être proportionnel à la distance entre A et B pour que l'ensemble médian $M(A,B)$ ne soit pas trop différent de A et B, si A et B sont proches l'un de l'autre, et notamment pour que $M(A,A)$ soit égale à A, ce qui par ailleurs garantit que A est bien le point fixe de f_1.

Des exemples de telles transformations (pour différents degrés de rotation) sont donnés en haut des figures 10 et 11. Notez que les ensembles médians exactes, $M(A,B)$, sont les mêmes à une rotation près. Les autres images intermédiaires, i.e. $M_{1/4}(A,B)$, sont par contre complètement différentes. Les rendus des attracteurs de mesure sont présentés en bas des figures respectives.

Les images résultantes dépendent continuement des

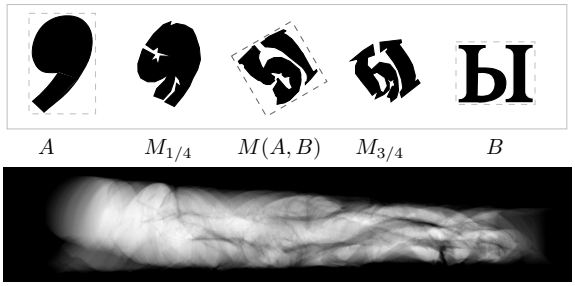


Figure 10: *L'ensemble médian est calculé à l'aide de la même procédure que l'exemple précédent (voir la figure 7), mais il a subit une rotation d'un angle proportionnel à la distance entre les ensembles A et B. L'image du bas montre le rendu de l'attracteur de mesure.*

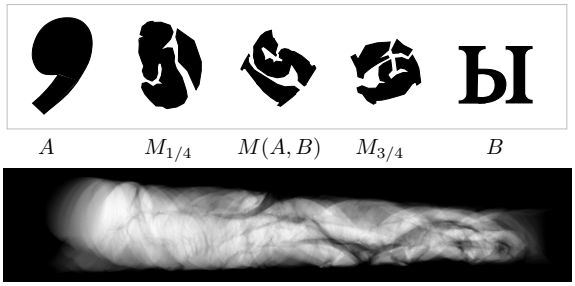


Figure 11: *Cet exemple est donné pour un autre degré de rotation. L'image du bas montre le rendu de l'attracteur de mesure.*

paramètres des transformations, i.e. des ensembles de condensation et du degré de rotation. Ainsi, en changeant les paramètres on obtient de remarquables animations.

5.2. Trois dimensions

Comme nous l'avons mentionné précédemment, le graphiste n'a pas besoin d'être mathématicien pour créer des objets ou des personnages vaporeux. Considérons un cas particulier d'IFS avec condensations, où le graphiste donne juste quelques esquisses et ne se préocupe ni des transformations ni des caractéristiques fractales. Alors, on peut utiliser les esquisses comme ensembles de condensation et ainsi déterminer un IFS pour lequel les transformations contractantes sont simplement des morphings.

Jusqu'à maintenant, tous les exemples étaient réalisés dans R^2, mais rien nous interdit d'opérer dans R^3. Supposons, que l'on souhaite obtenir une créature nuageuse représentant un taureau. La première étape consiste à réaliser un modèle polygonal du taureau (le modèle polygonal a été choisi pour des raisons de commodité par rapport au sujet à représenter). La seconde étape consiste à créer un modèle perturbé, voir figure 12 .

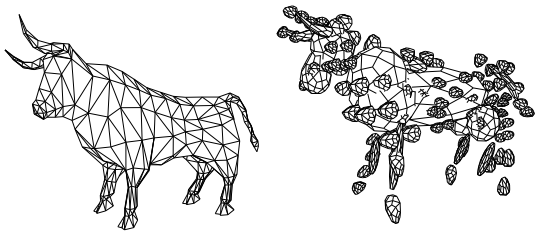


Figure 12: *Ces esquisses rapides donnerons naissance à une créature nuageuse.*

A noter que la deuxième étape de perturbation peut être réalisée de façon automatiquement ou semi-automatiquement. Puis les modèles sont voxélisés. Dans notre exemple, nous avons utilisé 256^3 voxels. Il est alors possible de calculer l'attracteur de mesure de l'IFS composé de deux transformations contractantes avec ensembles de condensation. La figure 13 présente le rendu de l'attracteur de mesure.

Figure 13: *Attracteur de mesure de l'IFS avec ensembles de condensation présentés par la figure 12. Les probabilités choisies sont respectivement* $\frac{2}{3}$ *et* $\frac{1}{3}$.

La différence entre les esquisses induit l'effet nebuleux. Si un voxel est présent dans chaque esquisse, il sera opaque et dense dans le rendu final, sinon il sera transparent. En affectant différentes probabilités aux transformations, on donne différents "poids " ou "importance' aux esquisses. Dans cet exemple les probabilités sont, respectivement, de $\frac{2}{3}$ et $\frac{1}{3}$. Ce choix force le rendu final à être plus proche de la première esquisse. Comme le graphiste n'est supposé devoir ajuster aucun paramètre, excepté les modèles 3D, la figure 13 est rendu à l'aide de l'algorithme déterministe, pour lequel l'IFS est constitué de trois transformations: une pour le modèle perturbé et deux autres identiques pour le modèle original du taureau. La figure 14 présente une version nuageuse du"vaisseau fantôme" que nous avons dèjà rencontré.

Puisque l'image résultante dépend continuement des paramètres des transformations, ici les ensembles de condensation (modèles 3D donnés en entrée), il est facile pour les graphistes, de créer des animations de manière classique. Par exemple, en utilisant un squelette d'animation, le taureau de nuage peut prendre vie. De plus, l'attracteur de mesure peut être importé dans le module Maya Fluid Effects. Ce qui permet de combiner notre méthode de création d'objets vaporeux avec une méthode basée sur une simulation physiques.

Figure 14: *Ce galion a été créé à partir de deux modèles. Le premier correspond au modèle du galion lui-même. Le deuxième a été créé en perturbant le premier de façon automatique à l'aide d'un script Python.*

6. Conclusions

Dans cet article nous avons présenté une généralisation du modèle IFS avec ensembles de condensation. Cette généralisation ouvre radicalement de nouveaux horizons pour la production d'images fractales. Toutes les implémentations antérieures apparaissent alors comme des cas particuliers de ce modèle. Par ailleurs, d'un point de vue purement esthétique, les IFS avec ensembles de condensation permettent un meilleur contrôle du résultat. Tout graphiste peut créer facilement un nuage fractal sans aucune connaissance de sa nature physique. Il peut également contrôler localement les formes sans affecter l'aspect globale. La création d'animation est facilitée par la propriété de continuité du modèle en fonction des paramètres d'entrées.

Au delà des nuages, nous pensons que d'autres phénomènes peuvent être rendus à l'aide des IFS avec ensembles de condensation. Par exemple, la fonction "top" de Barnsley et d'autres techniques offrent des possibilités de génération de textures. Par la suite, nous souhaitons examiner la possibilité de segmenter une image donnée pour en créer une version nuageuse.

References

[Bar88] BARNSLEY M.: *Fractals everywhere*. Academic Press Professional, Inc., San Diego, CA, USA, 1988.

[Bar06] BARNSLEY M. F.: *Superfractals*. Cambridge University Press, 2006.

[BD85] BARNSLEY M. F., DEMKO S. G.: Iterated function systems and the global construction of fractals. *Royal Society of London Proceedings Series A 399* (June 1985), 243–275.

[Dra05] DRAVES S.: The electric sheep screen-saver: A case study in aesthetic evolution. In *Applications of Evolutionary Computing, LNCS 3449* (2005), Springer Verlag.

[Dra06] DRAVES S.: The electric sheep and their dreams in high fidelity. In *The 4th International Symposium on Non-Photorealistic Animation and Rendering* (2006), ACM.

[EMP*02] EBERT D. S., MUSGRAVE K. F., PEACHEY D., PERLIN K., WORLEY S.: *Texturing & Modeling: A Procedural Approach, Third Edition (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, December 2002.

[HBSL03] HARRIS M. J., BAXTER W., SCHEUERMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Proceedings of the 2003 Annual ACM SIGGRAPH/Eurographics Conference on Graphics Hardware (EGGH-03)* (Aire-la-ville, Switzerland, July 26–27 2003), Mark W., Schilling A., (Eds.), Eurographics Association, pp. 92–101.

[Hut81] HUTCHINSON J.: Fractals and self-similarity. *Indiana University Journal of Mathematics 30*, 5 (1981), 713–747.

[IS00] IWANOWSKI M., SERRA J.: The morphological-affine object deformation. In *Proceedings of 5th International Symposium on Mathematical Morphology* (Palo Alto (USA), June 2000), Kluwer Academic Publishers 2000, pp. 81–90.

[Mey67] MEYERS P.: A converse to Banach's contraction theorem. *Journal of Research of the National Bureau of Standards 71B*, 2 (1967), 73–76.

[MYND01] MIYAZAKI R., YOSHIDA S., NISHITA T., DOBASHI Y.: A method for modeling clouds based on atmospheric fluid dynamics. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2001), IEEE Computer Society, p. 363.

[SDE05] SCHPOK J., DWYER W., EBERT D. S.: Modeling and animating gases with simulation features. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 97–105.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Insertion de détail dans des figures autosimilaires

Houssam Hnaidi, Eric Guérin et Samir Akkouche

LIRIS - Université Claude Bernard Lyon 1

Abstract

This paper deals with the approximation of natural objects using a fractal model equipped with a detail concept like wavelet transforms. The fractal model is a projected Iterated Function System (IFS) model. This model unifies the IFS model and a classical model used in computer graphics (free form representation with control points). The concept of wavelet transforms is used for adding a detail part to the fractal model. The approximation problem has a two steps formulation : a nonlinear fitting optimisation of the fractal model and a linear transform for the computation of the detail coefficients. We have used this approximation method to approximate two leaves border. Then, the detail concept has been used to transfer the geometric texture from one leaf to another, and also to amplify these geometric details.

Résumé

Cet article traite de l'approximation d'objets naturels grâce à un modèle fractal équipé d'un concept de détail comme celui présent dans les ondelettes. Le modèle fractal est un modèle IFS (Iterated Function System) projeté. Ce modèle unifie le modèle IFS et un modèle classique utilisés en infographie (déformation des objets grâce à des points de contrôle). Le concept des transformations par ondelettes est immité par l'ajout d'une partie de détail au modèle fractal. L'approximation se fait alors en deux phases : une phase d'optimisation non linéaire du modèle fractal et une phase de transformation linéaire qui permet de calculer les coefficients de détail. Nous avons employé cette méthode pour approximer la frontière de deux feuilles. Le concept de détail a ensuite été employé pour transférer la texture géométrique d'une feuille à une autre, ainsi que pour l'amplification de détails géométriques.

1. Introduction

La modélisation auto-similaire est un outil efficace pour produire des objets naturels qui possèdent d'emblée la propriété d'auto-similarité (arbres, montagnes, *etc.*). Les fractales, qui permettent de faire ce type de modélisation, ont été utilisées depuis longtemps pour synthétiser des plantes comme les fougères. Si les outils de synthèse sont très nombreux dans ce domaine, on ne peut pas en dire autant des outils d'analyse et d'approximation. Ces derniers doivent parvenir, à partir d'un objet naturel issu du monde réel, à extraire une représentation informatique qui le reproduise le plus fidèlement. Ces outils posent la problématique connue sous le terme de problème inverse. Jacquin a été l'un des pionniers dans la matière en introduisant une méthode de compression d'images basée sur un modèle fractal.

Dans un domaine connexe, les outils permettant une analyse multi-échelle des objets ont récemment connu une grande expansion, notamment par l'intermédiaire de ce qu'on l'on appelle les ondelettes. JPEG 2000 est un standard de compression d'image qui emploie cette théorie pour la compression des images.

A plusieurs reprises, des auteurs ont montré les liens qu'il pouvait y avoir entre la théorie des ondelettes et les fractales. Nous proposons dans ce papier de développer une méthode pour approximer les objets naturels basée sur un modèle fractal équipé d'un concept de détail comme celui présent dans les ondelettes. Notre modèle est capable de reconstruire ces objets de manière exacte et ce avec un paradigme multi-résolution. Nous avons appliqué ce modèle au transfert de textures géométriques de contours ainsi qu'à l'amplification de détails géométriques.

Dans une première partie, nous présentons un aperçu des travaux liés à cette étude. Dans une deuxième partie, nous introduisons quelques définitions pour la base théorique

de notre modèle. Notre contribution est introduite dans la troisième partie. Enfin, nous présentons quelques applications basées sur notre modèle.

2. Travaux liés

On peut classifier la géométrie fractale en deux domaines, les modèles stochastiques et les modèles déterministes. Le modèle stochastique d'un objet ou d'un phénomène est défini pour être un modèle où l'objet est représenté par un processus stochastique d'une ou plusieurs variables. Dans cette lignée, les travaux de Mandelbrot et Van Ness [Nes68] introduisent le concept de fBm (fractional Brownian motion). En revanche avec les modèles déterministes les phénomènes sont reproductibles, chaque instance de modèle génère une même figure. Parmi les modèles déterministes on peut citer les L-System [Lin68] qui ont plusieurs applications dans l'informatique graphiques [PH89, PL91, Smi84], ainsi que les IFS (Iterated Function System) [Bar88]. Fondés sur un formalisme mathématique rigoureux, les IFS sont un outil puissant pour l'analyse comme pour la synthèse d'objets fractals. Hutchinson [Hut81] fut le premier à les étudier en appliquant le théorème du point fixe aux ensembles de compacts d'un espace métrique complet. Barnsley [Bar93] a développé ce formalisme et l'a utilisé dans toute une série d'applications, entre autres, en infographie et en compression d'image.

Notre travail est basé sur une variante des IFS appelée IFS projeté [Zai98], ce modèle utilise le principe utilisé dans les formes à pôles comme Bézier par exemple. L'idée principale des IFS projetés est la séparation entre l'espace d'itération et l'espace de modélisation. Une forme à pôles consiste en deux parties: les points de contrôles et les fonctions de mélange. Les fonctions de mélange dans les IFS projetés deviennent des modèles IFS.

L'IFS projeté est efficace pour la modélisation des objets fractals, Zaïr [Zai98] est le premier à l'introduire et l'utilise pour la modélisation. Guérin [Gué02] a employé ce modèle pour l'approximation de courbes et de surfaces. Les avantages de ce modèle sont, premièrement qu'il s'agit de modèles simples à implémenter, et deuxièmement la séparation entre l'espace de modélisation qui est représenté par les points de contrôle, et l'espace d'itération qui est représenté par les IFS. L'utilisation des IFS projetés pour l'approximation de courbes et de surfaces pose le problème inverse qui est résolu par Guérin [GTB01] par la minimisation de la distance entre la courbe ou la surface originale et l'autre généré, mais cette méthode a besoin d'un modèle avec beaucoup de paramètres ce qui rend la méthode d'approximation complexe. Un autre inconvénient de l'IFS projeté est qu'il ne peut pas approximer tous les objets complexes.

Les modèles fractals possèdent la propriété d'autosimilarité. Cela signifie que l'objet est composé de parties qui lui ressemblent. La théorie des ondelettes [Mey87, Mal89] est utile pour étudier l'autosimilarité des signaux et des images, elle est utilisée pour la compression des images. Même si les ondelettes sont utiles et efficaces pour l'analyse et la synthèse des objets, la fonctionnelle utilisée (fonction d'ondelette et fonction d'échelle) dépend de l'application visée et non de l'objet lui-même.

L'autosimilarité est une propriété commune entre les IFS et les ondelettes, c'est pourquoi plusieurs personnes ont utilisés les IFS et les ondelettes ensembles pour analyser l'autosimilarité des objets [Dau88, Dau90, ET07].

3. La base théorique de notre modèle

Dans cette section nous expliquons la base théorique de notre modèle, en commençant par rappeler quelques définitions dans la théorie des IFS, puis dans les IFS projetés.

3.1. IFS

Ici on va mentionner les définitions principales dans la théorie des IFS.

Définition 1 (IFS) Soit $(\mathcal{X}, d)$ un espace métrique complet. Nous appelons IFS tout ensemble fini $\mathcal{T} = \{T_0, ..., T_{N-1}\}$ d'opérateurs contractants sur $\mathcal{X}$ [Bar88].

Définition 2 (théorème d'existence d'attracteur) Pour tout IFS $\mathcal{T}$, il existe un compact unique non vide A de $\mathcal{H}(\mathcal{X})$ tel que :

$$\begin{aligned} A &= \mathcal{T}A \\ &= T_0A \cup \ldots \cup T_{N-1}A \end{aligned}$$

A est appelé attracteur de $\mathcal{T}$ et sera noté $A(\mathcal{T})$.

Il est donc possible d'associer à tout IFS $\mathcal{T}$ un compact, appelé attracteur. Par définition, tout attracteur possède la propriété d'auto-similarité au sens large, c'est-à-dire :

$$A = T_0A \cup \ldots \cup T_{N-1}A$$

avec les T_i appartenant à une classe de fonctions contractantes.

Définition 3 (alphabet) Soit un IFS $\mathcal{T} = \{T_0, \ldots, T_{N-1}\}$.$\Sigma = \{0, ..., N-1\}$ sera appelé l'alphabet associé à $\mathcal{T}$.

En notant Σ^* l'ensemble des mots finis sur Σ, et Σ^w l'ensemble des mots infinis sur Σ, on a le résultat suivant (voir Barnsley [Bar88]) :

Soit $\theta = \theta_1\theta_2\theta_3 \ldots \in \Sigma^w$. Pour tout $\lambda \in \mathcal{X}$ la limite :

$$\lim_{j\to\infty} T_{\theta 1} \ldots T_{\theta j}\lambda$$

existe et est indépendante de λ.

On peut définir une fonction d'adressage, qui, à un mot infini sur Σ, associe un point de l'attracteur.

Définition 4 (fonction d'adressage) Soient $\mathcal{T}$ un IFS et Σ son alphabet associé. On appelle fonction d'adressage la fonction ϕ qui à mot infini de Σ fait correspondre un élément de $\mathcal{X}$ de la façon suivante :

$$\begin{aligned} \phi &: \Sigma^w \rightarrow \mathcal{X} \\ \theta &\mapsto \phi(\theta) = \lim_{j\to\infty} T_{\theta 1} \dots T_{\theta j} \lambda \end{aligned}$$

avec $\lambda \in \mathcal{X}$ quelconque.

La figure (1) donne un exemple d'IFS composé de quatre transformations affines qui sont chacune des combinaisons d'homothéties, translations et rotations.

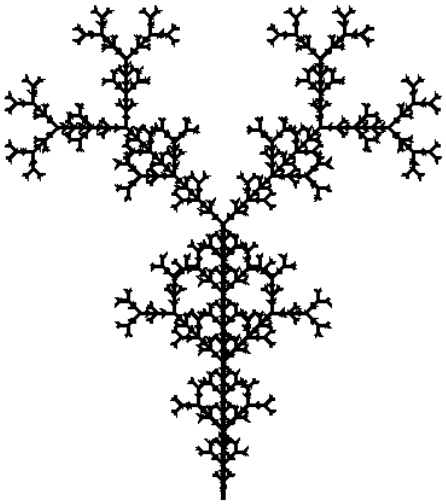


Figure 1: *Exemple d'IFS composé de transformations affines dans le plane*

3.2. IFS projeté

Pour fabriquer des fonctions de mélange fractales, il faut les définir en coordonnées barycentriques. Il faut donc choisir comme espace d'itération un espace barycentrique.

Définition 5 (espace barycentrique) Soit J un ensemble d'indices. L'espace barycentrique $\mathcal{B}^J$ associé à J est défini par :

$$\mathcal{B}^J = \left\{ (\lambda_j)_{j\in J} | \sum_{j\in J} \lambda_j = 1 \right\}$$

avec $\lambda_j \in \mathcal{R}$.

Il faut maintenant se fixer un semigroupe d'itération qui opère dans cet espace barycentrique. La solution la plus simple est d'utiliser des matrices dont les colonnes sont barycentriques.

Définition 6 (semigroupe de matrices barycentriques) Soit J un ensemble d'index. Le semigroupe de matrices barycentriques S_J associe à J est défini par :

$$S_J = \left\{ T | \sum_{j\in J} T_{ij} = 1, \forall i \in J \right\}$$

Grâce à ces choix, il est possible de ńprojeterż un attracteur d'IFS.

Définition 7 (attracteur d'IFS projeté) Soient J un ensemble d'indices, $\mathcal{T}$ un IFS constitué d'opérateurs de S_J, et $\mathcal{P} = (p_j)_{j\in J}$ une famille de points de contrôle. L'attracteur d'IFS projeté associé à $\mathcal{T}$ est défini par :

$$\begin{aligned} \mathcal{PA}(\mathcal{T}) &= \{\mathcal{P}\lambda | \lambda \in \mathcal{A}(\mathcal{T})\} \\ &= \mathcal{P}\phi(\theta) \\ &= \sum_{j\in J} \phi_j(\theta) p_j \end{aligned}$$

où $\mathcal{P}\lambda = \sum_{j\in J} \lambda_j p_j$

La figure (2) est un exemple de courbe fractale à pôle. Elle a été générée avec quatre points de contrôle et deux transformations.

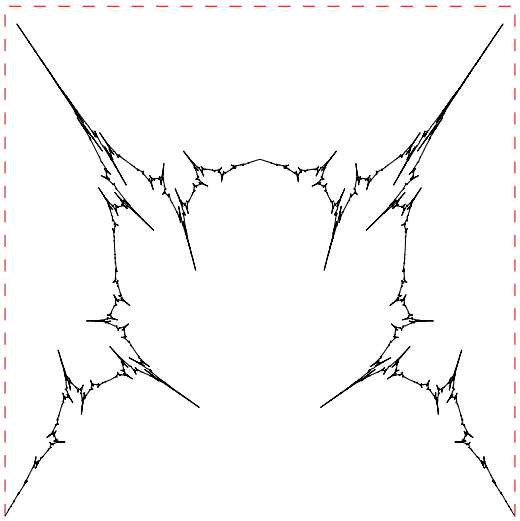


Figure 2: *Exemple de courbe généré avec IFS projeté*

3.3. Notre modèle

Afin de présenter une formule qui représente les détails insérés dans notre modèle fractals, on va commencer par présenter un moyen de visualiser les IFS projetés.

Définition 8 (visualisation d'attracteurs d'IFS projetés) Soient $(\mathcal{X}, d)$ un espace métrique complet, $\mathcal{T}$ un IFS, et $\mathcal{P}$ un vecteur de points de contrôle. Soit la suite définie par :

$$(\mathcal{S}_n)_{n\in\mathcal{N}} = \begin{cases} \mathcal{S}_0 & = \{\mathcal{P}\} \\ \mathcal{S}_{n+1} & = \mathcal{S}_n \mathcal{T}, \forall n \in \mathcal{N} \end{cases}$$

Alors, on a :

$$\lim_{n\to\infty} \mathcal{S}_n K = \mathcal{PA}(\mathcal{T})$$

avec $K \in \mathcal{H}(\mathcal{X})$.

$\mathcal{S}_n$ est un ensemble fini de polygones que l'on peut développer de manière récursive grâce à un arbre

$$\mathcal{S}_n = \mathcal{PT}^n = \left\{\mathcal{P}T_{\theta_1}\dots T_{\theta_n} \,|\, |\theta| = n\right\}$$

Afin de formaliser notre modèle on va écrire la suite précédente ainsi :

$$\mathcal{S}_n = \mathcal{P}T_\theta \quad \text{où} \quad T_\theta = T_{\theta_1}\dots T_{\theta_n} \quad \text{et} \quad |\theta| = n$$

Maintenant si on applique une seule transformation sur $\mathcal{S}_n$ alors on a :

$$\mathcal{S}_n T_i = \mathcal{P}T_\theta T_i \quad \text{où} \quad i \in \Sigma$$

en projetant le côté droit de l'équation précédente et en prenant la notation $\mathcal{P}T_\theta = \mathcal{P}_\theta$, on peut maintenant écrire :

$$\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i \qquad \text{où} \qquad i \in \Sigma$$

La figure (3) représente l'arbre utilisé pour visualiser cette formule. Cette figure montre le cas de deux transformations pour simplifier. On note P_ε le polygone initial de points de contrôle où ε représente le mot vide. Inspiré par les travaux

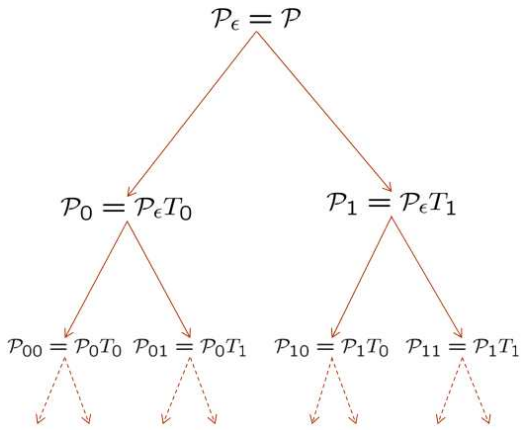


Figure 3: *Arbre de construction d'un IFS projeté*

de Tosan et al. [ET07] on peut ajouter une partie de détail sur la formule $\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i$ de cette manière :

$$\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i + \delta\mathcal{P}_\theta U_i \quad \text{où} \quad i \in \Sigma$$

Où $\delta\mathcal{P}_\theta$ est un vecteur de détail associé au polygone de points $\mathcal{P}_\theta$, et U_i (où $i \in \Sigma$) est une matrice que l'on appelle la matrice de déplacement de détail. La figure (4) représente l'arbre utilisé pour visualiser cette formule, aussi on utilise deux transformations et deux matrices de déplacement de détail pour simplifier le dessin. Comme on peut le remarquer sur la figure les détails sont ajoutés pour chaque niveau de l'arbre. Si on considère n points de contrôle, N transfor-

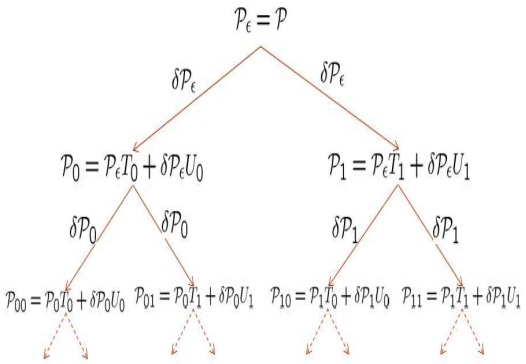


Figure 4: *Arbre de construction d'un IFS projeté équipé de détails*

mations, et on emploie l'espace $\mathcal{R}^k$, alors la dimension de la matrice T_i est $n \times n$, U_i est $n.(N-1) \times n$, $\mathcal{P}_\theta$ est $d \times n$, et $\delta\mathcal{P}_\theta$ est $d \times n.(N-1)$.

Le choix de la dimension des matrices de déplacement de détail et du vecteur de détail se base sur le fait que l'on cherche un système inversible.

En prenant la formule $\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i + \delta\mathcal{P}_\theta U_i$ pour tout $i \in \Sigma$ on a :

$$\begin{aligned} \mathcal{P}_{\theta 0} &= \mathcal{P}_\theta T_0 + \delta\mathcal{P}_\theta U_0 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \\ \mathcal{P}_{\theta N-1} &= \mathcal{P}_\theta T_{N-1} + \delta\mathcal{P}_\theta U_{N-1} \end{aligned}$$

Maintenant on peut écrire les formules précédentes sous forme matricielle :

$$(\mathcal{P}_{\theta 0}|\dots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta)\left(\begin{array}{c|c|c} T_0 & \dots & T_{N-1} \\ \hline U_0 & \dots & U_{N-1} \end{array}\right)$$

en utilisant la notation suivante :

$$R = \left(\begin{array}{c|c|c} T_0 & \dots & T_{N-1} \\ \hline U_0 & \dots & U_{N-1} \end{array}\right)$$

la formule devient :

$$(\mathcal{P}_{\theta 0}|\dots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta)\,R$$

la formule dite inverse est

$$(\mathcal{P}_\theta|\delta\mathcal{P}_\theta) = (\mathcal{P}_{\theta 0}|\dots|\mathcal{P}_{\theta N-1})\,R^{-1}$$

ainsi R peut être apparenté à un filtre de synthèse et R^{-1} à un filtre d'analyse utilisés dans la transformée en ondelette.

4. Application de notre modèle

Un domaine où l'on peut utiliser notre modèle est la reconstruction d'objets. Le but est de pouvoir reconstruire n'importe quel objet avec un minimum d'information de détails. Pour cela, une phase préalable d'optimisation de la

matrice R est nécessaire. Dans la suite, nous expliquons la méthode utilisée pour cette optimisation dans le cadre de la reconstruction de courbes.

4.1. Méthode d'optimisation pour les courbes

Dans cette partie, nous considérons qu'une courbe est un ensemble ordonné de points. La méthode d'optimisation est basée sur la minimisation de la distance entre la courbe originale et la courbe reconstruite avec notre modèle en prenant comme paramètres la position des points de contrôle et la matrice R. On note les points de la courbe originale par p_i où $i = 0 \dots n-1$ et n est le nombre de points initiaux. On note les points générés avec notre modèle par P'_i. Il est important de noter ici que notre modèle permettant de reconstruire parfaitement les données d'entrée, nous allons volontairement omettre une partie de l'information de détail lors de la reconstruction. La distance entre chaque doublon de points est définie par :

$$d_i = \left\| p_i - p'_i \right\| \quad \text{où} \quad i = 0 \dots n-1$$

Nous avons choisi la méthode de Levenberg-Marquardt [PFTV93] pour minimiser cette distance. Il s'agit d'une méthode de régression non linéaire basée sur la différentiabilité de la fonction à minimiser. Formellement, elle s'exprime de cette manière, soient :

- n points dans le plan $(x_i, y_i)_{i=0\dots n-1}$
- une fonction $y = f(x,a)$, où a représente une vecteur des paramètres.

Le but est de trouver le vecteur de paramètres a_{opt} qui approxime le mieux les données c'est-à-dire :

$$f(x_i, a_{\text{opt}}) \approx y_i)_{i=0\dots n-1}$$

Dans notre cas, chaque donnée d'entrée (x_i, y_i) va correspondre au couple $(i,0)$ et la fonction f sera définie par $f(i,a) = d_i$ où a est le vecteur de paramètres représentant dans notre cas la position des points de contrôle et les valeurs de la matrice R.

La fonction de mérite $\mathcal{X}^2(a)$ utilisée dans la méthode de Levenberg-Marquardt est la suivante :

$$\mathcal{X}^2(a) = \sum_{i=0}^{n-1} (0 - f(i,a))^2 = \sum_{i=0}^{n-1} (0 - d_i)^2$$

La méthode étant différentielle, nous devons calculer numériquement la dérivée de f par rapport aux paramètres a_k :

$$\frac{\partial f(i,a)}{\partial a_k} = \frac{f(i,a)_{a_k+\varepsilon} - f(i,a)}{\varepsilon}$$

avec $k = 0 \dots m-1$ où m est le nombre de paramètres.

Nous avons utilisé la notation $f(i,a)_{a_k+\varepsilon}$ pour dire que la valeur a_k du vecteur de paramètres a été modifiée de ε.

La minimisation s'effectue en deux étapes :

- Dans la première étape, on minimise la distance entre la courbe originale et la courbe construite avec notre modèle en utilisant les positions de points de contrôle et les coefficient des matrices $\mathcal{T}$ comme paramètres. Dans ce cas, aucune information de détail n'est utilisée pour reconstruire la courbe.
- Dans la deuxième étape, on minimise cette distance en prenant les coefficient des matrices $\mathcal{U}$ comme paramètres. Dans ce cas, une partie de l'information de détail seulement est utilisée, sinon la reconstruction serait parfaite et il n'y aurait rien à minimiser. En pratique, les trois derniers niveaux de détails sont omis.

Les détails utilisés dans l'optimisation sont produit par application de la formule d'analyse $(\mathcal{P}_\theta | \delta\mathcal{P}_\theta) = (\mathcal{P}_{\theta 0} | \dots | \mathcal{P}_{\theta N-1}) R^{-1}$ en commençant par les points initiaux jusqu'à arriver aux points de contrôle.

Les Figures (5,6) représentent deux exemples d'utilisation de notre modèle pour reconstruire deux courbes. La première représente la frontière d'un feuille de pomme, et la deuxième représente la frontière d'un feuille de cerise.

Pour les deux exemples, notre modèle est composé de quatre points de contrôle, deux transformations et deux matrices de détails. Aussi dans les deux exemples on a reconstruit les courbes en omettant les derniers trois niveaux de détails. Cela signifie que l'on utilise un huitième de l'information initiale.

4.2. Transfert de texture géométrique

Nous pouvons appliquer notre modèle au transfert de texture géométrique en utilisant la possibilité d'extraction de détails. Les coefficients de détails (à divers niveaux) peuvent être déplacés d'un objet à un autre. Sur un même objets, il peuvent être amplifiés, diminués. La puissance et la flexibilité de notre modèle pour l'analyse et la reconstruction nous donnent la liberté de l'utiliser pour diverses applications. Par exemple on optimise notre modèle avec un objet, puis on applique le modèle optimisé sur un autre objet, après on modifie les détails du premier objet en utilisant les détails du deuxième objet. La possibilité inverse est offerte : on optimise notre modèle avec le deuxième objet puis on modifie aussi les détails du premier objet.

Le figure (7) est un exemple de transfert de texture entre deux objets. le modèle qui est optimisé avec la courbe originale (courbe (a)) est utilisé pour l'opération de transfert de texture. En revanche la figure (8) utilise le modèle qui est optimisé avec la courbe qui contient la texture pour la même opération.

Le figure (9) donne un exemple d'amplification et diminution de détails et leur influence sur la texture.

5. Conclusion

Nous avons présenté une méthode pour approximer les courbes basée sur un modèle fractal équipé d'un concept de détail comme celui utilisé dans les transformations en ondelettes. Nous avons utilisé ce modèle pour approximer les courbes et pour le transfert et l'amplification de texture géométriques. L'avantage le plus important de notre modèle est qu'il est simple, car il utilise un nombre de paramètre plus faible pour approximer les courbes qu'avec un modèle fractal normal non doté de détail. Nos envisageons de développer notre modèle afin de traiter les surfaces de profondeur ainsi que les surfaces générales.

References

[Bar88] BARNSLEY M.: *Fractals everywhere*. Academic Press, 1988.

[Bar93] BARNSLEY M.: *Fractals everywhere (second edition)*. Academic Press Professional, 1993.

[Dau88] DAUBECHIES I.: Time-frequency localization operators: A geometric phase space approach. *IEEE Transactions on Information Theory 34*, 4 (1988), 605–612.

[Dau90] DAUBECHIES I.: The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory 36*, 5 (1990), 961–1005.

[ET07] E. TOSAN I. BAILLY-SALINS I. S. G. G. Y. W.: Modelisation iterative de courbes et surfaces : aspect multiresolution. In *Groupe de travail en Modelisation Geometrique journee de Valenciennes* (mars 2007), pp. 55–69.

[GTB01] GUÉRIN E., TOSAN E., BASKURT A.: Approximation de courbes et surfaces par une méthode fractale. In *CORESA 2001* (Dijon, nov 2001).

[Gué02] GUÉRIN E.: *Approximation fractale de courbes et de surfaces*. Thèse de doctorat, Université Claude Bernard Lyon 1, dec 2002.

[Hut81] HUTCHINSON: Fractals and self-similarity. *Indiana University Journal of Mathematics 30* (1981), 713–747.

[Lin68] LINDENMAYER A.: Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology 18*, 3 (March 1968), 300–315.

[Mal89] MALLAT S.: A theory for multiresolution signal decomposition: The waveletrepresentation. *IEEE Trans. Pattern Anal. Mach. Intell. 11*, 7 (1989), 674–693.

[Mey87] MEYER Y.: Les ondelettes. In *Contributions to nonlinear partial differential equations, Vol. II (Paris, 1985)*, vol. 155 of *Pitman Res. Notes Math. Ser.* Longman Sci. Tech., Harlow, 1987, pp. 158–171.

[Nes68] NESS. B. M. J. V.: Fractal brownian motions, fractal noises, and applications. *SIAM Rewiew*, 10 (1968), 422–437.

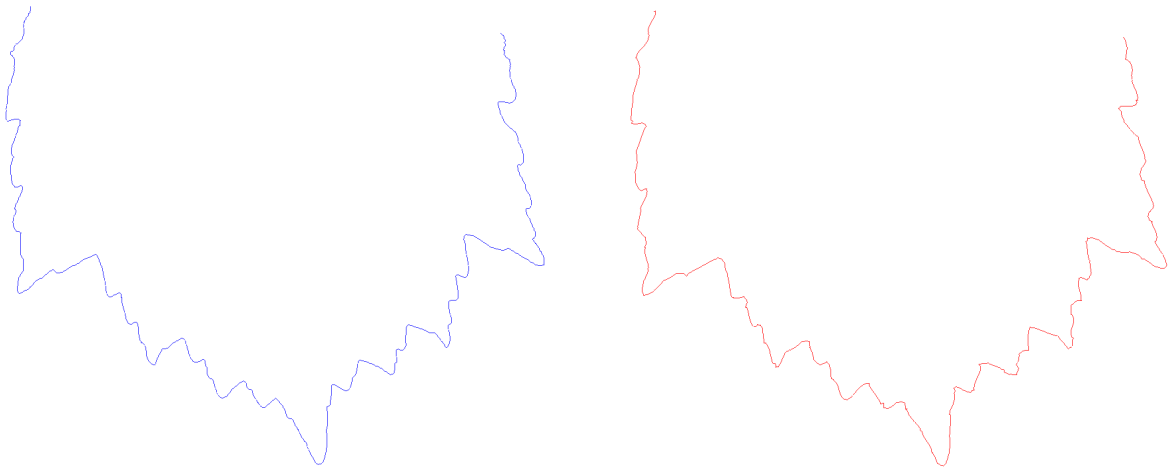
[PFTV93] PRESS W. H., FLANNERY B. P., TEUKOLSKY S. A., VETTERLING W. T.: *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1993, ch. Nonlinear Models.

[PH89] PRUSINKIEWICZ P., HANAN J.: Lindenmayer systems, fractals, and plants. *Lecture Notes in Biomathematics 75* (1989).

[PL91] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. Springer, October 1991.

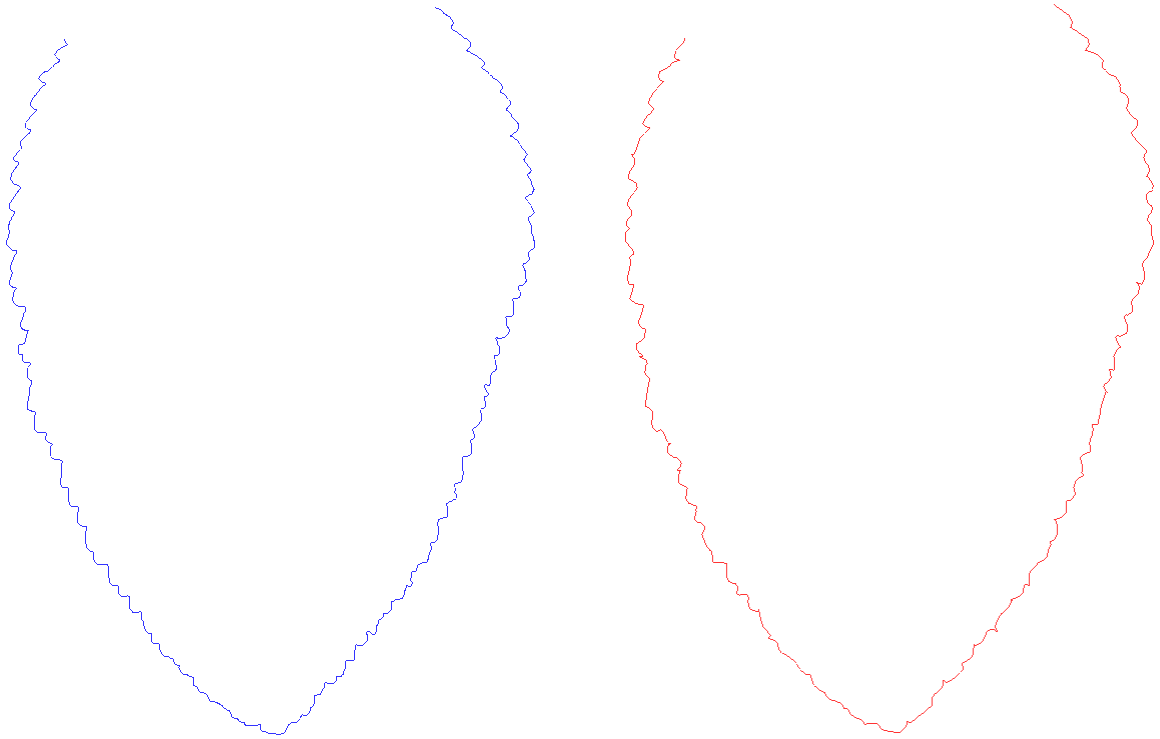
[Smi84] SMITH A. R.: Plants, fractals, and formal languages. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (July 1984), vol. 18, ACM Press, pp. 1–10.

[Zai98] ZAIR C. E.: *Formes fractales à pôles basées sur une généralisation des IFS*. Thèse de doctorat, Université Claude Bernard Lyon 1, jun 1998.



(a) (b)

Figure 5: *(a) La courbe originale de la feuille de pomme. (b) La courbe reconstruite en utilisant un huitième de l'information de la courbe originale.*



(a) (b)

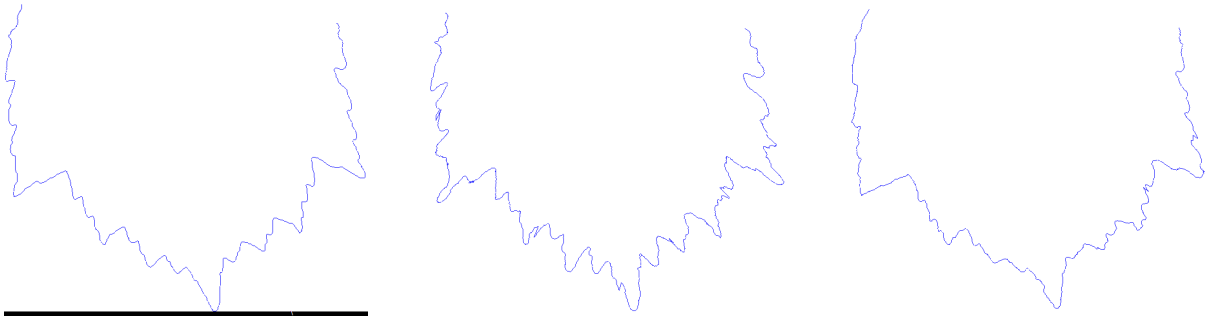
Figure 6: *(a) La courbe originale de la feuille de cerise. (b) La courbe reconstruite en utilisant un huitième de l'information de la courbe originale.*

(a) (b) (c)

Figure 7: *Transfert la texture en utilisant le modèle de la courbe originale. (a) La courbe originale. (b) La courbe qui contient la texture. (c) La courbe originale avec la texture de la deuxième courbe.*

(a) (b) (c)

Figure 8: *Transfert la texture en utilisant le modèle de la courbe qui contient la texture. (a) La courbe originale. (b) La courbe qui contient la texture. (c) La courbe originale avec la texture de la deuxième courbe.*



(a) (b) (c)

Figure 9: *(a) La courbe originale. (b) La courbe avec texture doublée. (c) La courbe avec texture diminuée.*

Affichage non aliasé pour l'illumination globale par voxels

Łukasz Piwowar[1] and Rémy Malgouyres[2]

[1]Institute of Computer Science, University of Wrocław,Wrocław 50-383,
Joliot-Curie 15, Poland
lpi@ii.uni.wroc.pl
[2]Univ. Clermont 1,
Laboratoire d'Algorithmique et Image (LAIC, EA2146),
IUT département informatique
B.P. 86, 63172 Aubière cedex, France
remy.malgouyres@laic.u-clermont1.fr

Abstract

Nous présentons une méthode d'affichage non aliasée basée sur le lancer de rayons pour l'illumination globale par voxels, qui est un alogorithme non biaisé, introduit dans [Mal02], [CM06], [ZFM06], pour résoudre l'équation d'illumination globale. Cette méthode hybride consiste à représenter le champs de radiance entrant par des sources luminauses ponctuelles virtuelles (VPLs), en composant la solution initiale en utilisant une structure de données spaciale formée de voxels. Le principal avantage est l'utilisation de la solution linéaire optimale pour résoudre les problèmes de visibilité avec cette structure de données de voxels.

1. Introduction

L'un des sujets aujourd'hui brûlant en synthèse d'images est l'illumination globale, et la recherche d'un moyen pour animer en temps réel un environnement virtuel photo-réaliste. L'objectif de l'illumination globale est de générer des images réalistes de scènes $3D$, qui soient correctes quantitativement concerant l'énergie émise par les différents éléments de la scène.

Les données en entrée sont la description de la scène et de la caméra, des maillages et les sources lumineuses courrament utilisées en synthèse d'images. Introduite par James Kajiya in 1986, l'équation d'illumination globale décrit le transfer d'énergie lumineuse d"un point à un autre comme une somme de la radiance émise et de la radiance réfléchie. Il en découle une équation d'équilibre énergétique appelée *équation d'illumination globale*. Les méthode d'illumination globale exactes tentent de calculer une solution numérique exacte pour cette équation. La solution ne dépend pas du point de vue.

Dans [Mal02], une approche entièrement nouvelle est proposée, ainsi qu'une discrétisation de l'équation d'illumination diffuse, basée sur une approximation des surfaces par des voxels. Cette discrétisation de l'équation conduit à un système linéaire dont les inconnues sont la radiance en chaque voxel. Une méthode pour résoudre ce système linéaire est présentée qui est beaucoup trop chère pour être praticable.

Dans [CM06], **une solution optimale en complexité (linéaire par rapport au nombre de rayons) est obtenue pour résoudre les problèmes de visibilité**. C'est un atout très fort pour cette approche comparativement au photon-maping ou a bidirectional path-tracing, qui sont fondées sur des algorithmes d'intersection rayon-objet. Cependant, l'affichage des voxels restait fortement aliasé et nécessitait de nombreux voxels pour produire un rendu acceptable, et la méthode parraissait toujours impraticable. Dans [ZFM06], on propose une version avec mémoire distribuée pour les scènes ayant de nombreux voxels.

Dans cet article, nous produisons une nouvelle méthode d'affichage utilisant un lancer de rayons progressif. L'affichage est non aliasé et produit à la fois des résultats précis et un bon rendu. Celà permet de réduire de beaucoup le nombre de voxels et le nombre de directions, faisant de notre méthode d'illumination globale, avec sa solution optimale unique en son genre pour la visibilité, un algorithme conpétitif pour l'éclairage indirect avec plusieurs réflexions successives. Comme seule la phase de lancer de rayons

dépend du point de vue, nous avons de réelles perspective d'animation temps réel de scènes statiques.

2. Illumination Equation and Classical Approaches

Let x and y denote two points in a scene. Let $B(x)$ denote the *radiosity* at the point x, the amount of the energy leaving x per unit of time per unit of solid angle (steradian) per unit of area. Let V denote the visibility function, defined by $V(x,y) = 1$ if y is visible from x in the scene and $V(x,y) = 0$ otherwise. Let r denote the distance between two points x and y. Let $\theta(x,y)$ denote the angle between the vector $\overrightarrow{xy}$ and the normal vector $\overrightarrow{n}$ at the point x (see Figure 1). If that angle is less then or equal to $\frac{\pi}{2}$, there is no interaction between x and y, and we define by convention $\cos\theta$ as equal to 0. The (continuous) diffuse illumination equation is as follows:

$$B(x) = E(x) + \rho(x) \int B(y) \frac{\cos\theta(x,y)\cos\theta(y,x)}{\pi r^2} V(x,y)\, dy$$

where

- A point re-emits only a fraction $\rho_d(x)$ of the energy it receives. Assuming that this factor does not depend on incoming or outgoing directions is known as the *ideal diffuse hypothesis*. The reflection coefficient ρ_d is less than 1.
- $\theta(y,x)$ is the angle between the vector $\overrightarrow{yx}$ and the normal $\overrightarrow{n'}$ to the surface at y.
- The π factor is a normalization term deriving from radiance considerations.

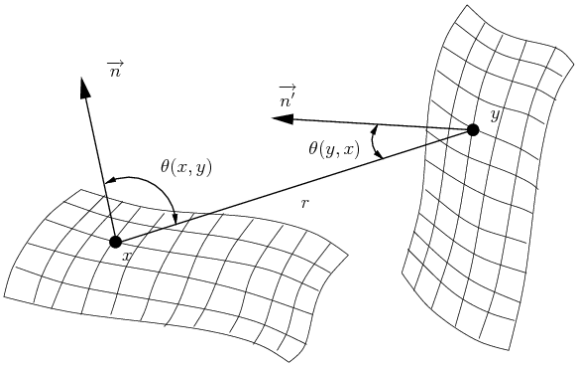


Figure 1: *Geometric relation*

An intuitive explanation of Equation (1) is: "the total power of light leaving a voxel x (the $B(x)$ term), is defined by two terms: the proper emittance of this voxel as a light source (the $E(x)$ term), and some fraction of the light it receives from its environment (the integral)".

Note that the equation presented here is a particular case, corresponding to the Lambertian model of reflection, of a more general global illumination equation with a *Bidirectional Reflectance Distribution Function* (*BRDF*). There is no theoretical obstruction to using our method with *BRDFs*, but only the diffuse case is fully implemented at this point.

We cannot present exhaustively all known approaches to global illumination in this paper for lack of space, but let us mention a few of them.

1. Radiosity with patches ([SP94]) decomposes the scene into some polygonal patches and makes the assumption that the radiosity is constant on each patch. The global illumination equation is then approximated by some linear system with good properties. The drawbacks of radiosity is the difficulty to make it work with *BRDFs*, and the high cost when the number of polygons increases (the visibility factor, called *form factor* between two patches is expensive to compute). Moreover, if we take too few patches, we cannot accurately display curved objects and mainly polygonal scenes are rendered.
2. Photon mapping ([Jen96], [Jen97], [Jen01]) is a method that traces random rays from light sources, and at each intersection, traces a new random ray with a probability that depends on the *BRDF*. This method accurately solves the illumination equation but indirect lighting is very expensive, especially since tracing a single ray, even with accelerating data structures, is far from constant time.
3. Bidirectional path tracing presents the same drawbacks as photon mapping.
4. Instant radiosity ([Kel97]) can be used in combination with other methods such as photon mapping for good looking display. Photons are used as virtual light sources in a raytracing phase. Instant radiosity is viewpoint-dependent. The method is very expensive on scenes lit primarily by indirect lighting, either with photon maps or quasi-random walk to distribute photons, several hundred frames may be required to get photons to arrive where they are needed.

Our method is from a family of methods that stores the outgoing energy in some spacial data structure (voxels in our case), which, similarly to radiosity, enables us to take advantage of dynamic programming, which is the only way to effectively simulate several successive reflections.

3. Voxel Based Approach

By a change of variable, using the correspondence between visible points from the point x and points on a sphere S centered at x, and the continuous diffuse illumination equation is equivalent to

$$B(x) = E(x) + \rho(x) \int_S B(y) \frac{\cos\theta}{\pi} \mathrm{d}\overrightarrow{\sigma}$$

In [Mal02], we simply discretize this equation by approximating the surfaces by a discrete surface, and also sampling the sphere to estimate the integral, by introducing a finite set of directions.

$$B(x) = E(x) + \rho(x) \sum_{\overrightarrow{\sigma} \in D} B(I(x, \overrightarrow{\sigma})) \frac{\cos\theta(x, I(x, \overrightarrow{\sigma}))}{\pi} \Delta\Omega(\overrightarrow{\sigma}) \quad (1)$$

where

- x is now a voxel;
- D is a set of discrete directions in space;
- $I(x, \overrightarrow{\sigma})$ is the first point y viewed from x in the direction of $\overrightarrow{\sigma}$ (as in a ray-object intersection);
- To quantify how much of an object is seen from a point, the term $\Delta\Omega(\overrightarrow{\sigma})$ is the fraction of a solid angle associated to the direction $\overrightarrow{\sigma}$.

This discrete equation is a large linear system with unknowns $B(x)$ but the $I(x, \overrightarrow{\sigma})$ depends on visibility and occlusions.

The way to obtain a discrete set of voxels from a continuous surface is described in [Mal94] for an implicit surface, and in [Mal02] for a mesh.

In [CM06], a solution of the linear system is obtained with optimal complexity. We use a slightly improved version of this method, by using bucket (or radix) sort (complexity $O(n)$) instead of quicksort. Thus final complexity is $O(N \times I \times D)$, where N is the number of voxels of the discrete surface, D is the number of directions used for sampling the sphere, and I is the number of iteration, or number of successive reflections used for indirect lighting. We emphasize that **the complexity is linear (with a small coefficient) with respect to the total number of rays traced**. In [PM08], an integer only discrete ray shooting is combined with the linear method for direct lighting of voxels to increase directions sampling from light sources.

The method has a potential as universal method for solving the general light transport problem with *BRDF*, but up to now only the diffuse case is implemented satisfactorily. Up until now, there was no satisfactory method for rendering the results. We solve this problem in this paper.

4. Optimal Complexity for Visibility

To solve Equation (1), a converging iterative method similar to the one used with classical patch-based radiosity can be used: it usually relies on Jacobi or Gauss-Seidel relaxation. If we consider the linear equation under its form $B = E + M.B$, where B is a vector of elements and M a matrix of factors, some properties of M ensures the sequence $B_{n+1} = E + M.B_n$ converges toward a limit, which is a solution of the discrete linear system. Roughly speaking, this is a transcription of light gathering, each iteration going a step further in light re-emission. Convergence is expected since light is progressively absorbed. Technically, each iteration consists in propagating packets of energy between mutually visible voxels.

Given a direction vector $(a,b,c) \in \mathbb{Z}^3$ with $a \geq b \geq c$, a notion of a $3D$ line has been proposed [DR95], as the set of points $(x,y,z) \in \mathbb{Z}^3$ such that

$$\mu \leq cx - az < \mu + \omega \text{ and } \mu' \leq bx - ay < \mu' + \omega'$$

where $\mu, \mu', \omega, \omega'$ are integers. Other cases can be deduced by symmetry. It is noteworthy that under this definition, a $3D$ discrete line represents the intersection between two discrete planes, each being the orthogonal extrusion of a $2D$ discrete line included in one of the coordinate planes. The connectivity is related to the thickness ω and ω' of these $2D$ lines. If the two $2D$ projections are naïve (resp. standard), the $3D$ line is called naïve (resp. standard).

Proposition 1 Let us denote by $\mathbb{Z}^3_*$ the set $\mathbb{Z}^3 \backslash \{(0,0,0)\}$. Given an integer vector $\overrightarrow{v} \in \mathbb{Z}^3_*$, the set $\mathbb{Z}^3$ can be partitioned into $3D$ discrete lines, whose direction vector is $\overrightarrow{v}$ (see Figure 3). Moreover, given a voxel $x \in \mathbb{Z}^3$, finding out which $3D$ discrete line in the partition the point x belongs to can be done in constant time.

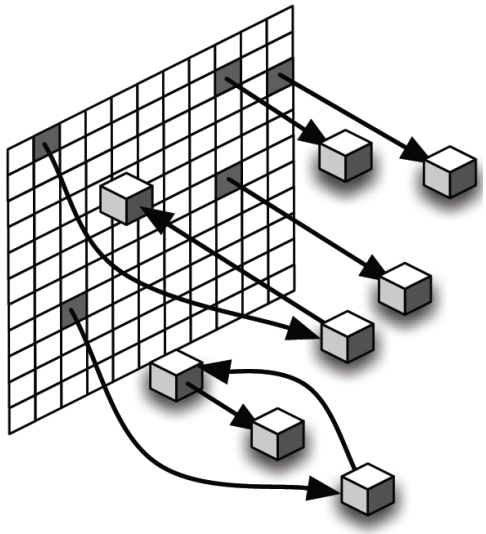
Now, solving our linear system given by Equation (1) by the Gauss-Seidel relaxation amounts to transferring energy from one voxel x to the first voxel y visible from x in a direction σ, for some finite set of sample directions σ. We can assume w.l.o.g that σ has integer coordinates. Now, if we consider some fixed direction σ and a partition of the voxel space $\mathbb{Z}^3$ into discrete lines parallel to σ, then the voxels of the discretized surface (say mesh or implicit surface) that lie on the same discrete line can be arranged in an ordered list. In this ordered list, the first visible voxel is the next voxel in the list (see Figure 2). So, once the lists are sorted, we can propagate the energy in linear time $O(N)$.

Now, the idea is that by going over the set of all surface voxels in a lexicographic order (lexicographic orders are precomputed by radix sort), we can build all the lists in linear time $O(N)$ (see Figure 3). We do this for each of the D sample directions and for each of the I Gauss-Seidel iteration, and we get a numerical solution of Equation (1) in time $O(N \times I \times D)$.

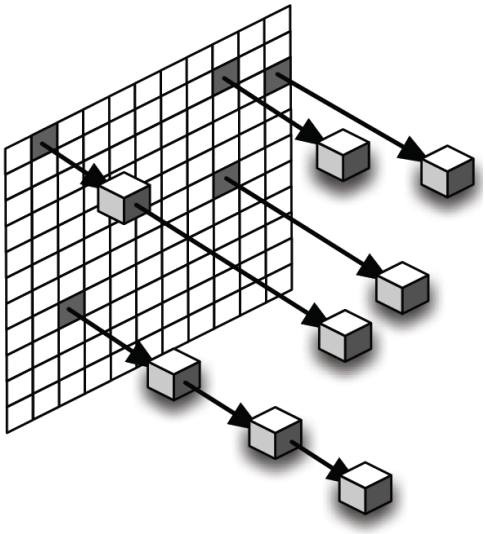
5. Display Method Using RayTracing

To compute the final image we use raytracing and use the voxels as a light sources (or $\mathbb{VPL}\sim$) in the way somewhat inspired by instant radiosity method [Kel97]. Of course, the principle must be substantially modified to be adapted to voxels. The intensity of the voxels is proportional to their radiosity as obtained in the output of the method of [CM06]. However, to take into account the nature of radiosity (power per unit of steradian per unit of area) we select the voxels *randomly* according to a probability proportional to their *area*, as defined below, and multiplied by a solid angle.

Each connected component χ of the continuous surface (e.g mesh) separates the space into two connected components, C and $\overline{C}$ (Jordan theorem), where $\overline{C}$ is the component which contains the viewpoint. We consider O the set of all



(a) Unsorted lists of voxels



(b) Sorted lists of voxels

Figure 2: *Lists of voxels obtained by intersection with discrete lines*

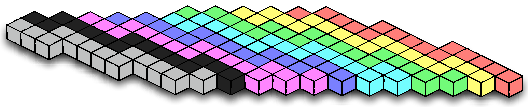


Figure 3: *The lexicographic on voxels coincides with linear order on rays*

the voxels $(i, j, k) \in Z^3$ which are contained in C (i.e. voxels inside an object or outside the scene). The discretization of the surface is the boundary F of O, which is the set of all voxels in O which are 6−adjacent to (i.e. have a common face with) a voxel in the complement $\overline{O}$ of O.

For a voxel $x \in F$, the element of area $\Delta A(x)$ is the sum, for all 6−neighbors y of x in $\overline{O}$, of the dot product $\overrightarrow{xy}.\overrightarrow{N}$, where $\overrightarrow{N}$ is the normal vector at the point x.

$$\Delta A(x) = \sum_{y \in N_6(x) \cap \overline{O}} \overrightarrow{xy}.\overrightarrow{N}$$

So, we make a raytracing phase by tracing rays from the viewpoint through the pixels. For each pixel, we compute the ray-object intersection point I that we must shade. In order to shade the point I, we use the sample voxels y, randomly selected with probability proportional to $\Delta A(y)$, as point light sources with intensity, or *energy contribution*:

$$C_y(I) = B(y) \frac{\cos\theta(I,y)\cos\theta(y,I)}{||I-y||^2} V(I,y)$$

In fact, we select two random samples sets of voxels: one for light sources and one for non light source voxels. The reason is the light sources (non-zero emittance) emit much more power and must be sampled more finely. We denote by $\mathcal{L}$ the set of all voxels with non-zero emittance (light source voxels) and $\mathcal{NL}$ the set of all non light-source voxels. We select a set $V_d \subset \mathcal{L}$ of light source sample voxels. and a set $V_i \subset \mathcal{NL}$ of non light source sample voxels. We denote $n_d = |V_d|$ the number of light source sample voxels and $n_i = |V_i|$ the number of non light source sample voxels, The total power at the pixel (or equivalently at I) is computed as :

$$\frac{1}{2\pi}\left(\frac{TAD}{n_d}\cdot\sum_{y\in V_d} C_y(I) + \frac{TAI}{n_i}\cdot\sum_{y\in V_i} C_y(I)\right)$$

where:

$$TAD = \sum_{x\in\mathcal{L}} \Delta A(y)$$

$$TAI = \sum_{x\in\mathcal{NL}} \Delta A(y)$$

are the total areas of light source surfaces $\mathcal{L}$ and non light source surfaces $\mathcal{NL}$.

We can add some specular effects at the point I using a local illumination model. We implemented successfully the *BRDF* model of ([AS00]).

6. Implementation

We use progressive raytracing that allows the user to get approximate results after a few seconds. For each frame, part of all the Virtual Point Lights (*VPL*) is taken into account. Thus at each instant the preview image is an approximation of the energy of the final solution. The output converges to the final solution and computation can be stopped by the user or automatically (by using a difference threshold between successive frames). Since we use the same set of *VPLs* for all the pixels (and possibly for all viewpoints in the same static scene), no noise is noticeable (the only problem is hard shadow edges with disappear after a sufficient number of *VPLs* is taken into account).

To create a proper *VPL* order, we select the voxels randomly according to a probability proportional to their *area*. To do this, we place all the voxels in a table, compute a random number between 0 to the sum of the areas ΔA, and use a binary search to select the right voxel. Conformly to theory of statistics, we do not reject a voxel after selection if selected more then once.

In the raytracing step, at each frame, for each pixel we compute the contribution of the next 16 of the *VPLs* to the final picture.

We add features like anti-aliasing and depth of field without additional cost, by slightly modifying the ray's position and/or direction. For both effects we use stratified sampling computed once for each frame and used by each pixel.

7. Results

Tests were done on an *Intel Core 2 Duo 6300* (1.86GHz). Only one core was used. Our computer has 2GB of memory (but generally less then 1GB was used). We used a completely home-made $CPU-$only raytracer, using $KD-$trees with a simplified surface-area heuristics. Our implementation of raytracing has many shortcomings, and so has our implementation of our voxel method. In particular, both are $CPU-$only and use no threads. Statistics about test scenes are shown in Table 1 and Table 2. Detailed number of rays per second are shown in Table 4.

Table 1: *Test scenes statistics (1)*

Scene	Voxels	LPL	SR	LR	Size	TT
Bunny1	146355	5	35	22	1024x768	558
Bunny30	146355	5	35	22	1024x768	2486
Bunny116	146355	5	35	22	1024x768	7742
Sponza	236754	7	35	26	1024x768	29185
Labirynth	11803	9	35	22	512x512	207
Emission	36062	4	8	22	512x512	112
Shadow	26547	4	22	18	512x512	147
Geometric	21074	4	28	18	512x512	208

Table 2: *Statistics for different kinds of rays (2)*

Scene	ST	LT	RTT	SRays	LRays	CRays
Bunny1	93	404	52	70.5e6	807.6e6	9.45e6
Bunny30	93	404	1980	70.5e6	807.6e6	297e6
Bunny116	93	404	7236	70.5e6	807.6e6	1 133e6
Sponza	274	960	27755	211.4e6	1 181e6	2 030e6
Labirynth	12	68	124	22e6	183e6	48.5e6
Emission	5	50	56	3.7e6	284e6	8.7e6
Shadow	4	23	119	3.2e6	141e6	29.3e6
Geometric	11	20	176	8.1e6	109e6	43.5e6

These results inspire us two kinds of considerations.

1) With comparable implementations, the number of rays per second is by far greater for discrete rays than for continuous rays. This shows that a $GPU-CPU$ implementation of the linear discrete method will, beyond reasonable doubt, outrun $KD-$tree based ray-tracing. Moreover, the discrete method's performance is less sensitive to surfaces' complexity.

2) The discrete method combined with display by classical ray-tracing provides both accurate global illumination and good rendering. When coupled with a good $GPU-CPU$ implementation of ray-tracing, it will be a very competitive global illumination method.

Table 3: *Detailed column names for Tables 1 and 2*

Name	Description
Scene	Scene name
Voxels	Number of voxels
LPL	Light path length (number of successive reflexions +1)
SR	Shooting discrete sphere radius
LR.	Linear method discrete sphere radius
Size	Width and height in pixels of the output image
TT	Total runtime (in seconds)
ST	Voxel shooting time (in seconds)
LT	Linear method propagation time (in seconds)
RTT.	Ray-tracing display time (in seconds)
CRays	number of continous rays (raytracing)
SRays	number of discrete rays (shooting)
LRays	number of discrete rays (linear method)
DSR	Discrete shooting rays
DLR	Discrete linear method rays
CR	Continuous rays (classical ray-object intersection)

Table 4: *Rays per second*

Scene	DSR	DLR	CR	CR/DSR	CR/DLR
Bunny30	758 064	1 999 010	150 105	19.8%	7.5%
Sponza	771 533	1 230 208	73 129	9.5%	5.9%
Labirynth	1 833 333	2 691 176	391 373	21.2%	14.5%
Emission	670 556	5 636 372	154 896	23%	2.7%
Shadow	703 332	5 886 649	246 006	35%	4.2%
Geometric	740 149	5 453 033	247 269	33.4%	4.5%

Conclusion

Our experimental results show that Voxel Based Global Illumination can be competitive for exact simulation and has a potential in the future for real-time animation of static scenes. It has a better complexity than any other known accurate method for indirect lighting. Planned future work include plugging the method into some real-time *GPU* raytracer, developing effective techniques for non-static scenes and including non-lambertian model, including caustics in the simulation.

References

[AS00] Michael Ashikhmin and Peter Shirley. An anisotropic Phong BRDF model. *Journal of Graphics Tools: JGT*, 5(2):25–32, 2000.

[CM06] Pierre Y. Chatelier and Remy Malgouyres. A low-complexity discrete radiosity method. *Computers & Graphics*, 30:37–45, February 2006.

[DR95] Isabelle Debled-Rennesson. *Étude et reconnaissance des droites et plans discrets*. PhD thesis, Université Louis Pasteur, Strasbourg, 1995. PhD thesis.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien.

[Jen97] Henrik Wann Jensen. Rendering caustics on non-Lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997.

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the ACM SIGGRAPH Conference (SIGGRAPH-97)*, pages 49–56, New York, August 3–8 1997. ACM Press.

[Mal94] R. Malgouyres. *Une définition des surfaces de Z^3*. PhD thesis, Université Clermont 1, 1994.

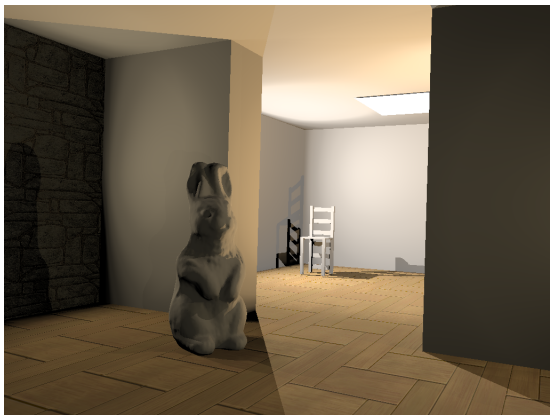
[Mal02] Rémy Malgouyres. A discrete radiosity method. In Achille J.-P. Braquelaire, Jacques-Olivier Lachaud, and Anne Vialard, editors, *DGCI*, volume 2301 of *Lecture Notes in Computer Science*, pages 428–438. Springer, 2002.

[PM08] Łukasz Piwowar and Rémy Malgouyres. Combining shooting and gathering for voxel global illumination. Technical report, Université Clermont 1, 2008. Submitted for publication.

[SJ00] B. Smits and H. Jensen. Global illumination test scenes. Technical report, University of Utah, 2000. Technical report.

[SP94] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, San Francisco, 1994.

[ZFM06] Rita Zrour, Fabien Feschet, and Rémy Malgouyres. Parallelization of a discrete radiosity method using scene division. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (2)*, volume 4276 of *Lecture Notes in Computer Science*, pages 1213–1222. Springer, 2006.



(a) 16 *VPL*

(b) 480 *VPL*

(c) 1856 *VPL*

Figure 4: *Bunny test scene, progressive rendering after 1, 30 and 116 frames*

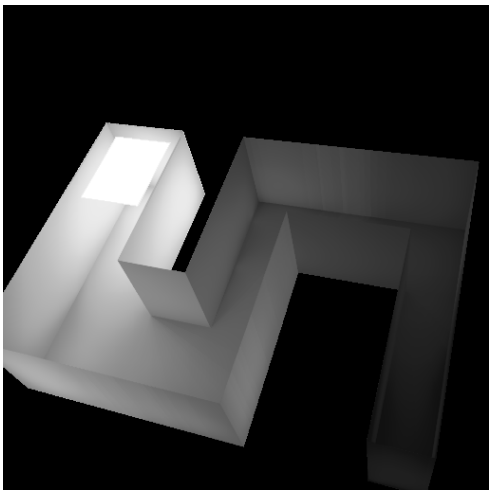


Figure 5: *Labirynth test scene showing extreme indirect lighting*

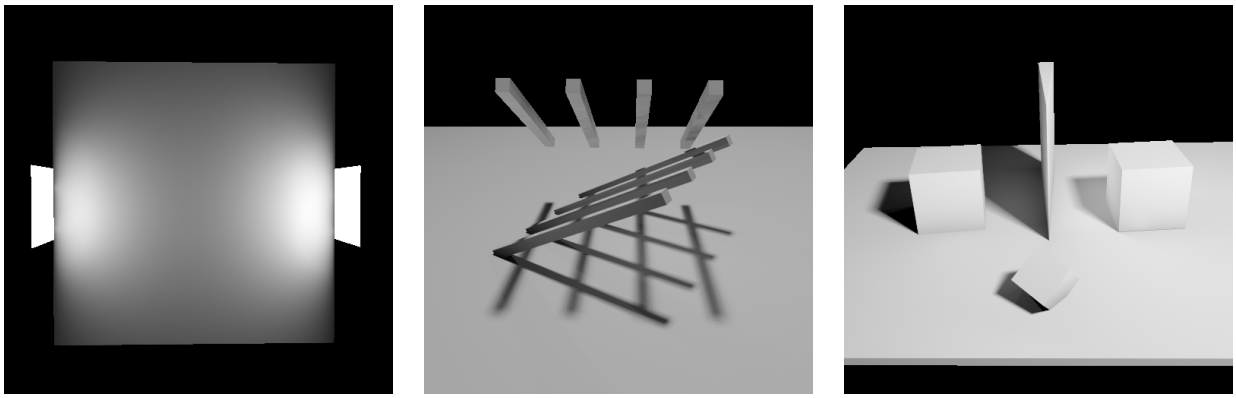


Figure 6: *Three of the Global Illumination Test Scenes ([SJ00]) (emission test, shadow test, and geometric accuracy)*

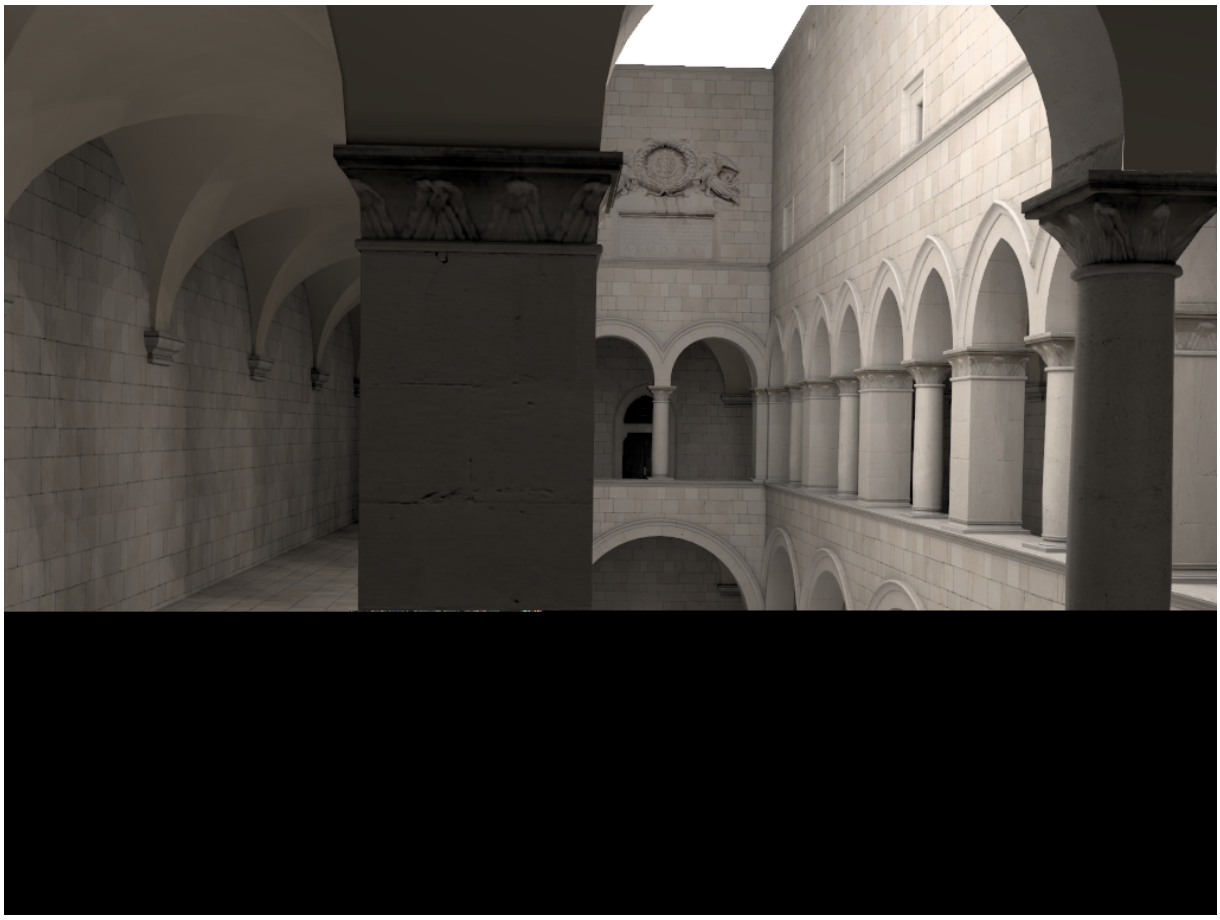


Figure 7: *The Sponza Atrium test scene*

Figure 8: *The Sponza Atrium: voxel view (without tone mapping)*

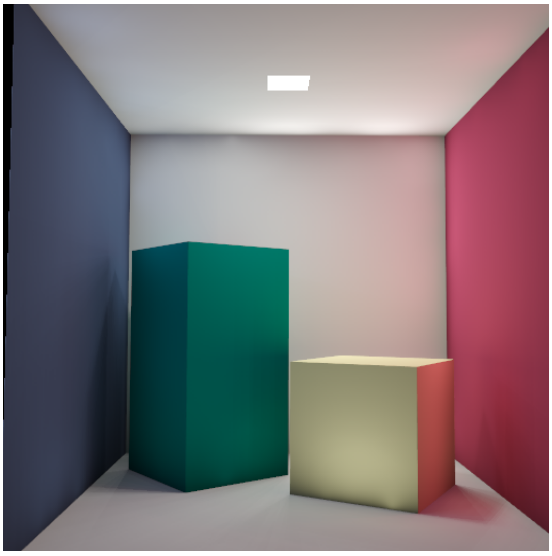


Figure 9: *The Cornell box*

Algorithme d'intersection entre un rayon et un carreau de Bézier par quasi-interpolant bilinéaire

Authors here

Abstract

We present a novel Bézier patch ray intersection algorithm. As opposed to most of the current techniques, that are either based on ray implicitization or on envelop construction, we use recent results on linear approximation of Bézier curves to efficiently determine surface parts that may contain a potential intersection. The major advantage of our approach lies in the abilitity to directly compute the maximum deviation between a Bézier patch and its bilinear approximation (known as quasi-interpolant). Since the deviation is function of the second order difference of the control net, by combining Bézier clipping and iterative construction of quasi-interpolants, our algorithm extracts surface parts that contain potential intersections and tend to the tangent plan to the surface at the intersection point. At last, our approach is also characterized by its simplicity because it only requires projections and linear transformations.

Nous présentons un nouvel algorithme d'intersection entre un rayon et un carreau de Bézier. A la différence de la plupart des techniques existantes, qui sont soit basées sur des méthodes d'implicitisation du rayon ou soit sur des techniques de construction d'enveloppe de la surface, nous utilisons de récents résultats concernant l'approximation linéaire des courbes de Bézier, pour extraire efficacement les portions de surfaces pouvant contenir une intersection potentielle. Le principal avantage de notre approche est de pouvoir directement calculer le maximum de déviation entre un carreau de Bézier et une approximation bilinaire (polygone quasi-interpolant). Comme, la déviation est fonction de la différence seconde des points de contrôle de la surface, en combinant Bézier clipping et construction itérative de polygones quasi-interpolants, notre algorithme permet d'isoler des portions de carreaux de taille strictement décroissante qui tendent vers le plan tangent à la surface au point d'intersection. Enfin, notre approche est également caractérisée par sa simplicité puisqu'elle ne nécessite seulement que des projections et des transformations linéaires.

1. Introduction

Depuis plusieurs décennies, les techniques de lancer de rayon ont été intensément étudiées et ont trouvé de nombreuses applications dans les domaines de la synthèse d'images, de la visualisation et de l'animation. Parmi les nombreuses tâches effectuées par un algorithme de lancer de rayons, la plus importante est celle qui consiste à déterminer efficacement l'intersection entre un rayon et un objet d'une scène virtuelle ainsi que la normale à la surface en ce point. En effet, les scènes utilisées aujourd'hui sont souvent composées de plusieurs centaines de milliers d'objets complexes, souvent modélisés par des NURBS ou des surfaces de subdivision, et nécessitent plusieurs millions de lancers de rayons (primaires ou d'ordre supérieur, rayons d'ombres, etc) pour lesquels il est impératif de pouvoir rapidement et précisément determiner les potentielles collisions avec les objets de la scène. La plupart des travaux liés aux techniques de lancer de rayons pour des objets de forme libre peuvent être séparés en deux catégories : ceux utilisant directement la surface et ceux construisant des enveloppes de ces surfaces pour simplifier les calculs d'intersection. Parmi les nombreuses solutions proposées pour lancer des rayons sur des surfaces paramétriques, comme par exemple des surfaces de Bézier, des NURBS ou des sufaces de subdivision, les algorithmes de Nishita et son extension aux NURBS proposée par Campagna et al. [CSS97] isolent les parties de surfaces ne contenant pas d'intersection et les évacuent par Bézier clipping. Benthin *et al.* ont proposé plusieurs stratégies pour optimiser l'algorithme de Nishita. Plus spécifiquement, ils ont montré qu'un choix minutieux entre des algorithmes bien con-

nus, des structures de donnés appropriées, une architecture et son implémentation pouvait conduire à une significative accélération des temps de calculs.

D'autres travaux s'intéressent aux techniques de construction de boites englobantes de la surface, comme celle proposée par exemple par Kobbelt *et al.* [KDS98] pour les surfaces de subdivision. Malheureusement, cet algorithme est relativement lent et la construction des boites englobantes peut parfois être incorrecte, comme le soulignent Peters et Wu [PW04] en introduisant le concept d'enveloppe adaptative. Les travaux concernant la construction d'enveloppe de courbe ou de surfaces s'appuient tous sur des résultats théoriques concernant les majorants et minorants de la déviation entre une courbe et son polygone de contrôle. En s'appuyant sur ces majorants, introduits par Lutterkort [Lut99] et Reif [Rei00], Peters *et al.* ont proposé les *Sleves* (subdividable linear efficient variety enclosure). Dans [WP04], Wu et Peters combinent les concepts introduits par Kobbelt [KDS98] et Lutterkort [Lut99] pour construire des enveloppes de surfaces de subdivision de Loop. Müller *et al.* ont également proposé un algorithme original de lancer de rayon sur des surfaces de subdivision de Loop, connu sous le nom de Shaolin (shadow of line). Leur approche consiste à projeter l'ombre d'un rayon sur une approximation de la surface limite pour déterminer l'intersection du rayon et de la surface.

Dans cet article, nous nous intéressons au problème du calcul du ou des points d'intersection entre un rayon et une surface paramétrique tout en conservant une attention particulière aux besoins inhérents d'un algorithme de lancer de rayon, *i.e.* minimiser la complexité en temps et en opérations, réduire la charge mémoire, et si possible, effectuer des calculs de même nature (projections et applications matricielles dans notre cas). Notre algorithme ne nécessite aucune construction géométrique comme pour le cas d'enveloppe, et les opérations effectuées à chaque itération ne consistent qu'en une seule projection et plusieurs applications linéaires (Bézier clipping et construction du quasi-interpolant). Notre algorithme peut être appliqué à des carreaux de Bézier de bi-degré (d_1, d_2) quelconque, ce qui permet notament de pouvoir l'étendre aux surfaces de subdivisions de Catmull-Clark et Doo-Sabin, dans le cas de maillages réguliers.

Le reste de ce papier suit la structure suivante : la section 2 présente les récents résultats de Zhang et Wang sur la majoration de la déviation entre une courbe de Bézier et son polygone quasi-interpolant. Dans la section 3, nous étendons ces résultats au produit tensoriel de courbes de Bézier pour définir un majorant de la déviation entre un carreau de Bézier et son quasi-interpolant. La Section 4 est consacrée à la présentation détaillée de notre algorithme. Nous présentons nos conclusions et perspectives de recherche dans la section 5.

2. Quasi-interpolant d'une courbe de Bézier et déviation maimum

Reif et Lutterkort ont indépendamment proposé des majorants de la déviation entre les courbes splines et leurs polygones de contrôle [Lut99, Rei00]. Ces majorants sont exprimés à partir de la différence du second ordre des points de contrôle. Nous reprenons dans cet article les notations utilisées par Reif [Rei00]. Soit $\mathbb{P}^d$ l'espace des polynomes de degré inférieur ou égal à d sur l'intervalle $[0,1]$. On note $B^d = \left[B_0^d, \ldots, B_d^d\right]$ le vecteur ligne des polynomes de Berstein de degré d, $H^d = \left[H_0^d, \ldots, H_d^d\right]$ les fonctions linéaires par morceaux entre les points de contrôle définis pour les valeurs paramétriques $t_j^d = j/d$, $j = 0, \ldots, d$ et $L^d = B^d - H^d$ leur différence.

Tout polynome g peut être écrit dans la base de Bernstein-Bézier comme $g = B^d P$, où $P = [P_0, \ldots, P_d]^T \in \mathbb{R}^{d+1}$ est un vecteur colonne de réels.

Le polygone de contrôle correspondant $h = H^d P$ est une fonction linéaire par morceaux ayant pour valeurs P_j aux points de contrôle ayant pour abscisse t_j^d. Le vecteur de la μ-ième différence des points de contrôle P est noté $\Delta^\mu P = \left[\Delta_0^\mu P, \ldots, \Delta_{d-\mu}^\mu P\right]$.

Pour tout $t \in [0,1]$, P correspondant aux $d+1$ points de contrôle, P^* correspondant aux points de Greville, et Δ^2 la différence seconde des points de contrôle, le théorème 2.2 de Reif [Rei00] établit :

$$\left\| L^d(t) P \right\|_\infty \leq \left\| L^d(t) P^* \right\|_\infty \left\| \Delta^2 P \right\|_\infty \tag{1}$$

Le théorème 3.1 de Nairn *et al.* [NPL99] propose un majorant similaire de la déviation entre une courbe et son polygone de contrôle, défini par :

$$\left\| L^d(t) P \right\|_\infty \leq N_\infty(d) \left\| \Delta^2 P \right\|_\infty \tag{2}$$

Le coefficient de Nairn, noté $N_\infty(d)$ dépend du degré d de la courbe, et est défini par :

$$N_\infty(d) = \frac{\lfloor d/2 \rfloor \lceil d/2 \rceil}{2d} \tag{3}$$

Dans [ZW06], Zhang et Wang ont proposé un majorant plus fin de la différence entre une courbe de Bézier et son polygone de contrôle quasi-interpolant, qui correspond à une ligne brisée dont les points sont construits par des combinaisons convexes des points de contrôle. Pour une courbe $C^d(t)$ de degré d, les points de contrôle du quasi-interpolant Q^d, notés Q_i^d sont définis par :

$$Q_0^d = P_0, Q_n^d = P_n,$$
$$Q_k^d = \frac{P_{k-1} + 2P_k + P_{k+1}}{4}, k = 1, \ldots, d-1.$$

Ce qui peut être écrit sous forme matricielle comme :

$$Q^d = M_d P \tag{4}$$

Par exemple, pour une courbe de Bézier cubique, M_3 est définie par :

$$M_3 = \frac{1}{4}\begin{pmatrix} 4 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 4 \end{pmatrix} \tag{5}$$

En utilisant la notation de Reif, la déviation maximum entre une courbe de Bézier d et son polygone quasi-interpolant $Q^d(t)$ peut être écrite comme :

$$\left\| C^d(t) - Q^d(t) \right\|_\infty = \left\| \left(B^d(t) - H^d(t) M_d \right) P \right\|_\infty$$

Zhang et Wang démontrent dans [ZW06] que la déviation entre une courbe $C^d(t)$ et son polygone quasi-interpolant $Q^d(t)$ est majorée par :

$$\left\| \left(B^d(t) - H^d(t) M_d \right) P \right\|_\infty \leq Z_\infty(d) \left\| \Delta^2 P \right\|_\infty \tag{6}$$

Le coefficient de Zang, noté $Z_\infty(d)$ est également fonction du degré d de la courbe, et peut être explicitement exprimé en fonction du coefficient de Nairn $Z_\infty(d)$ par la relation suivante :

$$Z_\infty(2) = \frac{1}{16}, Z_\infty(d) = N_\infty(d) - \frac{1}{4}, d \geq 3 \tag{7}$$

Zhang et Wang rappellent que les valeurs de $N_\infty(d)$ et $Z_\infty(d)$ pour $d = 2, \ldots, 8$ sont respectivement définies par

$$\left(\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{3}{5}, \frac{3}{4}, \frac{6}{7}, 1 \right) et \left(\frac{1}{16}, \frac{1}{12}, \frac{1}{4}, \frac{7}{20}, \frac{1}{2}, \frac{17}{28}, \frac{3}{4} \right).$$

La figure 1(a) présente un exemple de courbe avec son polygone de contrôle et son quasi-interpolant.

3. Quasi-interpolant d'un carreau de Bézier

Dans le cas d'un produit tensoriel de polynomes de bi-degré $d = (d_1, d_2)$, tout polynome $g(u,v)$ peut s'écrire dans la base de Bernstein-Bézier comme $g(u,v) = B^{d_1}(u) P \left(B^{d_2}(v) \right)^T$, où P est une matrice $(d_1+1) \times (d_2+1)$ de points de contrôle. Le polygone de contrôle correspondant est une fonction bilinéaire par morceaux $h(u,v) = H^{d_1}(u) P \left(H^{d_2}(v) \right)^T$. La déviation entre g et h est exprimée par un opérateur linéaire $L^d(u,v)$ agissant sur les matrices de points de contrôle et défini par :

$$L^d(u,v) = B^{d_1}(u) P \left(B^{d_2}(v) \right)^T - H^{d_1}(u) P \left(H^{d_2}(v) \right)^T$$

En utilisant les théorèmes 2.1 et 2.2 de [Rei00], Reif étend ses résultats au produit tensoriel :

$$\left\| L^d(u,v) P \right\|_\infty \leq \left\| L^{d_1} P^* \right\|_\infty \left\| \Delta_1^2 P \right\|_\infty + \left\| L^{d_2} P^* \right\|_\infty \left\| \Delta_2^2 P \right\|_\infty \tag{8}$$

Ce qui peut être également exprimé en utilisant les coefficients de Nairn sous la forme :

$$\left\| L^d(u,v) P \right\|_\infty \leq N_\infty(d_1) \left\| \Delta_1^2 P \right\|_\infty + N_\infty(d_2) \left\| \Delta_2^2 P \right\|_\infty \tag{9}$$

En d'autre termes, dans le cas du produit tensoriel, le majorant de la déviation peut être décomposé comme une somme des majorants le long de chaque dimension. Les différences secondes sont alors définies par :

$$\left\| \Delta_1^2 P \right\| = \max_{0 \leq i \leq d} \left\| \Delta^2 P_i \right\|_\infty \text{ et}$$
$$\left\| \Delta_2^2 P \right\| = \max_{0 \leq i \leq d} \left\| \Delta^2 (P_i)^T \right\|_\infty$$

Où $P_0, \ldots, P_d$ représente les vecteurs colonne de P. Dans le cas de carreaux de Bézier de bi-degré $d = (d_1, d_2)$, le polygone quasi-interpolant Q^d est défini par :

$$Q^d = M_{d_1} P (M_{d_2}) T \tag{10}$$

Où $P = (P_{ij})$ est une matrice $(d_1+1) \times (d_2+1)$ de points de contrôle, M_{d_1} et M_{d_2} sont des matrices carrées $(d_1+1) \times (d_1+1)$ et $(d_2+1) \times (d_2+1)$, respectivement.

La déviation maximum entre une surface de Bézier bivariée $S^d(u,v)$ de bi-degré $d = (d_1, d_2)$ et son polygone quasi-interpolant $Q^d(u,v)$ est définie par :

$$\left\|S^d(u,v) - Q^d(u,v)\right\| = \left\|S^d(u,v) - (M_{d_1} P M_{d_2}^T)(u,v)\right\| =$$
$$= \left\|B^{d_1}(u) P \left(B^{d_2}(v)\right)^T - H^{d_1} M_{d_1} P M_{d_2}^T \left(H^{d_2}(v)\right)^T\right\|$$
$$= \left\|B^{d_1}(u) P \left(B^{d_2}(v)\right)^T - H^{d_1} M_{d_1} P \left(\left(H^{d_2}(v)\right) M_{d_2}\right)^T\right\|$$

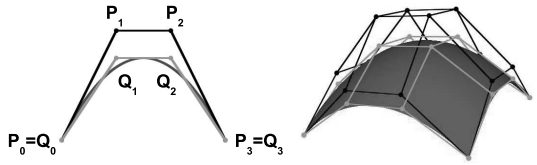
Puisque le maximum de la déviation $\|(B-HM)P\|_\infty$ est majoré (équation 6), nous pouvons utiliser l'extension de Reif au produit tensoriel pour determiner un majorant de la déviation entre un carreau de Bézier et son polygone quasi-interpolant, défini par :

$$\left\|(S^d - Q^d)(u,v)\right\|_\infty \leq Z_\infty(d_1) \left\|\Delta_1^2 P\right\|_\infty + Z_\infty(d_2) \left\|\Delta_2^2 P\right\|_\infty \quad (11)$$

Pour les carreaux de surfaces de Bézier bicubiques sur l'intervalle $[0,1]^2$, *i.e.* $d_1 = d_2 = 3$ et $Z_\infty(3) = 1/12$, on obtient un nouveau majorant de la déviation, défini par :

$$\left\|(S^3 - Q^3)(u,v)\right\| = \frac{1}{12}\left(\left\|\Delta_1^2 P\right\|_\infty + \left\|\Delta_2^2 P\right\|_\infty\right) \quad (12)$$

La figure 1(b) présente un carreau de Bézier bicubique et son polygone quasi-interpolant.



(a) (b)

Figure 1: *Polygone quasi-interpolant. a) Courbe de Bézier cubique. b) Carreau de Bézier bicubique.*

4. Algorithme d'intersection entre un rayon et un carreau de Bézier

Notre approche partage de nombreux points communs avec la méthode de Nishita, notamment l'évacuation des zones ne pouvant contenir d'intersection par Bézier clipping. Notre algorithme procède de la manière suivante : on considère en entrée un (ou plusieurs) carreau de Bézier pour lequel on construit le polygone quasi-interpolant. On projète ensuite le polygone quasi interpolant sur un plan orthogonal au rayon et on détermine les carreaux paramétriques pouvant contenir une intersection. On évacue ensuite les parties non désirés par Bézier clipping et itérons cette opération jusqu'à ce que le maximum de la différence seconde des points de contrôle, notée ε devienne inférieur à une valeur définie par l'utilisateur. Quand ε atteint la précision désirée, la portion de carreau de Bézier peut être considérée comme plane et on peut ainsi déterminer l'intersection entre ce plan et le rayon.

Nous savons qu'à chaque itération, la surface se situe dans une enveloppe de largeur 2ε autour du quasi-interpolant. On pourrait construire cette enveloppe pour essayer de déterminer une zone susceptible de contenir une intersection. Cependant, la construction d'enveloppe nécessite la création et la maintenance d'une structure géométrique supplémentaire et alourdit sensiblement la taille des données utilisées par l'algorithme. Une approche plus fine consiste à ne pas construire une ε-enveloppe autour du quasi-interpolant mais autour du rayon, ce qui conduit à un cylindre. La détection d'une intersection potentielle pourra donc être ramenée à vérifier quels sont les carreaux du quasi-interpolant qui intersectent le cylindre de rayon ε centré autour du rayon. Cette technique présente l'avantage majeur de ne nécessiter aucune création d'enveloppe car on peut directement obtenir une équation implicite pour un cylindre, ce qui facilite grandement les tests d'intériorité nécessaires à la détection des collisions. Cependant, on peut encore réduire la complexité des calculs (principalement des calculs de distance) en projetant le polygone de contrôle sur un plan orthogonal au rayon. Il reste ensuite à vérifier si un point, qui correspond à la projection du rayon, appartient à la projection des carreaux de surfaces bilinéaires qui constituent le quasi-interpolant.

Pour déterminer efficacement si un point est susceptible d'appartenir à la projection d'une surface bilinéaire, nous avons trois cas gérer. Pour verifier l'appartenance d'un point au projeté d'une surface bilinéaire, nous utilisons le fait que les bords d'un carreau bilinéaire sont des droites. Par conséquent, leur projection sera également des droites, ou éventuellement des points, si le bord est parallèle au rayon. La projection des bords d'un carreau bilinéaire conduit à trois type de polygones : convexe, non convexe non auto-intersectant et non convexe auto-intersectant, comme l'illustre la figure 2. Dans les deux derniers cas, la projection d'un carreau bilinéaire n'est plus un polygone avec un nombre de côtés finis, ce qui amène à des cas particuliers et des calculs trop spécifiques. Pour se ramener à un seul type de polygone et éviter de fastidieux calculs, on considère l'enveloppe convexe du projeté. Ceci simplifie grandement l'algorithme car déterminer l'appartenance d'un point à un polygone convexe à trois ou quatre sommets en deux dimensions est un problè particulièrement bien connu. En revanche, on discriminera moins facilement les carreaux du quasi-interpolant qui contiennent effectivement des intersection potentielles, ce qui se traduira par une augmentation du nombre de carreaux à traiter.

Chaque carreau bilinéaire projeté correspond à un intervalle paramétrique $I \in [0,1]^2$. Pour chaque carreau paramétrique, on sait construire un nouveau polygone de contrôle par Bézier clipping [Far02]. De plus, comme toutes les valeurs paramétriques possibles sont identiques à chaque

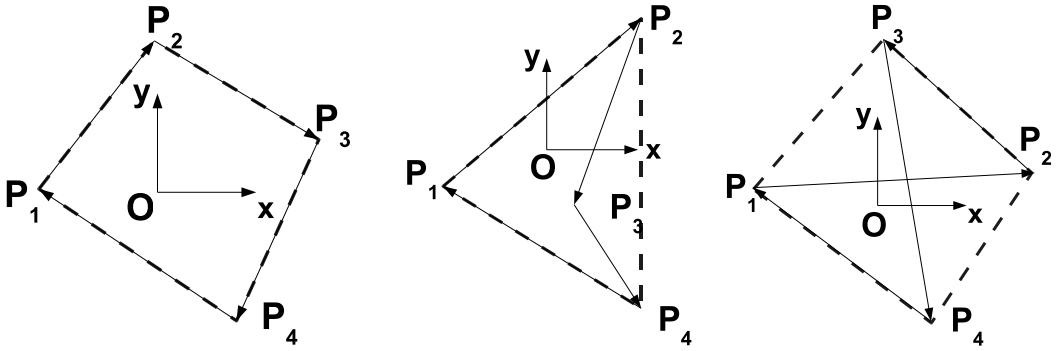


Figure 2: *Polygone résultat de la projection des bords d'un carreau de surface bilinéaire. a) Convexe. b) Non convexe non auto-intersectant. c) auto-intersectant.*

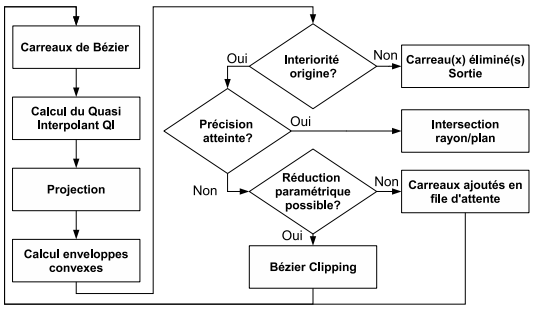


Figure 3: *Diagramme de l'algorithme d'intersection entre un rayon et un carreau de Bézier*

itération, on peut calculer une fois pour toutes les différentes matrices de clipping à l'initialisation. Les nouveaux points de contrôle C_j après Bézier clipping sont définis par :

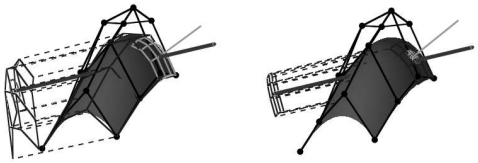
$$C_j = \sum_{j=0}^{d} P_j B_j^d(t) \tag{13}$$

où les points P_j sont les points de contrôle et $B_j^d(t)$ les polynomes de Bernstein-Bézier;

Pour construire le polygone de contrôle correspondant à la courbe de Bézier sur l'intervalle paramétrique $[t_0, t_1]$ on applique successivement une première fois la matrice de clipping sur l'intervalle $[0, t_1]$ puis une seconde fois, avec un changement de variable, pour obtenir le polygone de contrôle sur l'intervalle $[t_0, t_1]$. Dans le cas de carreaux de Bézier, on applique donc quatre matrices de clipping pour construire le nouveau maillage de contrôle.

Dans le cas d'un carreau de bi-degré $d = (d_1, d_2)$ on effectue donc $d_1 \times d_2$ tests d'appartenance d'un point à un polygone par itération de l'algorithme. Au cours des premières itérations, on peut détecter une intersection potentielle pour plusieurs intervalles paramétriques, notamment dans le cas de carreaux de surface présentant une forte courbure autour du point d'intersection. Dans ce cas, on empile les valeurs paramétriques correspondantes pour chaque carreau dans une pile d'intervalles paramétriques restant à traiter. Pour chaque nouvel intervalle paramétrique stocké dans la pile, on construit un nouveau maillage de contrôle qui est traité individuellement dès qu'aucun autre patch n'est plus en cours de traitement.

La figure 4 illustre le résultat de notre algorithme durant les trois premières itérations et le résultat final. Sur chaque figure, on peut voir la surface limite, le polygone de contrôle en noir, le quasi interpolant du carreau considéré en gris sombre et son projeté, le maillage de contrôle après clipping qui sera utilisé à la prochaine itération en gris clair, et une approximation du point d'intersection et de la normale à la surface en ce point. Durant la première itération, trois intersections potentielles sont détectées. Les figures 4(b) et 4(b) ne montrent que le résultat de l'algorithme pour le premier intervalle paramétrique. Quand une intersection est déjà calculée, un test supplémentaire permet de déterminer si on doit appliquer à nouveau l'algorithme sur les carreaux restants. Pour ce faire, on construit le polygone quasi-interpolant du carreau et on projète ses sommets le long du rayon. Si tous les sommets projetés sont situés après le point d'intersection calculé précédemment, le carreau est totalement situé derrière le point d'intersection et ne nécessite aucun traitement supplémentaire. Comme seule l'intersection la plus proche de l'origine du rayon nous intéresse, le carreau est évacué, comme l'illustre la figure 5. Dans le cas contraire, c'est-à-dire quand il peut exister une intersection située en amont du point d'intersection déjà calculé, on applique l'algorithme pour déterminer une nouvelle intersection. L'intersection finalement conservée est l'intersection la plus proche de l'origine du rayon.



(a) Itération 1 (b) Itération 2

(c) Itération 3 (d) Résultat final

Figure 4: *Intersection et normale calculée. Résultat final obtenu en cinq itérations pour une précision* $\varepsilon = 10^6$.

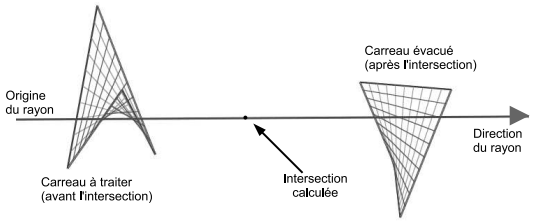


Figure 5: *Intersections multiples : sur la gauche, le carreau situé avant l'intersection doit être traité, le carreau sur la droite est situé après l'intersection et peut être évacué.*

5. Conclusions and perspectives

Nous avons présenté un nouvel algorithme d'intersection entre un rayon et un carreau de surface de Bézier. Notre algorithme utilise des résultats récents concernant la construction d'une approximation linéaire d'une courbe de Bézier et tire profit du plus fin majorant de la déviation maximale entre un carreau de Bézier et son quasi-interpolant. Combiné à avec des méthodes de clipping connues, il permet de déterminer l'intersection et la normale à la surface en ce point sans avoir à résoudre de systèmes d'équations, évitant ainsi le recours aux méthodes de Newton et certains problèmes de mauvaises intersection [CSS97]. De plus, aucune construction d'enveloppe comme [KDS98] ou toute autre structure géométrique supplémentaire n'est nécessaire puisque le quasi-interpolant est le résultat d'une application linéaire sur le polygone de contrôle. Enfin, les opérations effectuées à chaque itération ne sont que deux types : des projections ou des applications linéaires, ce qui ouvre la porte à l'implémentation de notre algorithme sur processeurs graphiques.

Les perspectives de recherche sont encore nombreuses et concernent plusieurs aspects : l'extension de notre algorithme aux surfaces de subdivision irrégulières (car on peut déjà appliquer notre algorithme pour des surfaces de subdivision régulières comme Catmull-Clark ou Doo-Sabin, qui convergent vers des surfaces B-Splines), l'amélioration de la réduction de l'intervalle paramétrique pour réduire le nombre d'itération et l'utilisation des GPUs pour gérer ces calculs, favorisés par des opérations de types similaires et des tailles de données adéquates (matrices 4x4, vecteurs lignes ou colonnes 4-dimensionnels dans le cas de surfaces bicubiques).

References

[CSS97] CAMPAGNA S., SLUSALLEK P., SEIDEL H.: Ray tracing of spline surfaces: Bézier clipping, chebyshev clipping, and bounding volume hierarchy - a critical comparison with new results. *The Visual Computer 13*, 6 (August 1997), 265–282.

[Far02] FARIN G.: *Curves and surfaces for CAGD a practical guide*. Morgan-Kaufmann, 2002.

[KDS98] KOBBELT L., DAUBERT K., SEIDEL H.: Ray tracing of subdivision surfaces. In *in In Proc. of 9th Eurographics Workshop on Rendering* (1998), pp. 69–80.

[Lut99] LUTTERKORT D.: *Envelopes of Nonlinear Geometry*. PhD thesis, Purdue University, December 1999.

[NPL99] NAIRN D., PETERS J., LUTTERKORT D.: Quantitative bounds on the distance between a polynomial piece and its bézier control polygon. *Computer Aided Geometric Design 16*, 7 (1999), 613–631.

[PW04] PETERS J., WU X.: Sleves for planar spline curves. *Computer Aided Geometric Design 21*, 6 (2004), 615–635.

[Rei00] REIF U.: Best bounds on the approximation of polynomials and splines by their control structure. *Computer Aided Geometric Design 17*, 6 (2000), 579–589.

[WP04] WU X., PETERS J.: Interference detection for subdivision surfaces. *Computer Graphics Forum 23*, 3 (2004), 577–584.

[ZW06] ZHANG R., WANG G.: Sharp bounds on the approximation of a bézier polynomial by its quasi-control polygon. *Computer Aided Geometric Design 23* (2006), 1–16.

Animation événementielle de structures topologiques : application à la construction de chenaux

Pierre-François Léon[1], Xavier Skapin[1] et Philippe Meseure[1]

[1]Université de Poitiers - Laboratoire SIC

Abstract

Ce papier présente un modèle permettant de décrire l'évolution temporelle de structures topologiques en 2D. Pour cela, nous proposons un modèle d'animation générant cette évolution à partir d'un scénario. Le modèle est constitué de trois parties : le modèle structurel autorisant la représentation temporelle de la topologie et de la géométrie, un modèle événementiel qui vise à détecter les modifications topologiques et s'occupe de la cohérence entre la topologie et la géométrie et un modèle sémantique représentant l'évolution en termes de modifications élémentaires et chargé de gérer l'historique des diverses entités du modèle. Pour illustrer l'effectivité de ce modèle, nous proposons une application en géologie à travers la modélisation des phénomènes de sédimentation et d'érosion nécessaire à la création de chenaux.

This paper presents a model which describes the temporal evolution of topological structures in 2D. For this purpose, we propose an animation model that generates this evolution from a scenario. The model is composed of three parts : A structural model allowing the temporal representation of the topology and the geometry, an event model which aims at detecting topological modifications and takes care of the consistency between the topology and the geometry, and a semantic model representing the evolution as a series of elementary modifications and managing the history of the various entities of the model. To show the efficiency of our model, we propose an application in geology through the modeling of sedimentation and erosion, two phenomena which are essential in the channel creation process.

1. Introduction

De nombreuses sciences expérimentales (biologie, botanique, géologie, etc.) sont confrontées à des structures naturelles très élaborées dont les lois de formation sont complexes et souvent mal maîtrisées. Un modèle permettant de représenter l'évolution de ces structures, de contrôler cette évolution, de remettre en cause les phénomènes et de fournir l'historique des diverses entités se révélerait un outil appréciable pour comprendre les causes, la formation et les évolutions possibles d'une structure. Dans cet article, nous proposons une première étape dans la conception d'un tel outil. Il s'agit d'une méthode générale d'animation de structures topologiques 2D à partir de scénarii. Plus précisément, nous proposons un modèle qui représente l'évolution d'une subdivision de l'espace par application d'un ensemble de modifications topologiques à des instants donnés. À tout moment de l'animation, la structure et la géométrie du système sont définies (afin de générer les images successives de l'animation). De plus, l'évolution individuelle de chaque entité du modèle est représentée de façon compréhensible pour l'utilisateur. En outre, ce modèle garantit en particulier la cohérence entre le modèle géométrique et le modèle topologique, i.e. il ne suffit pas que le résultat soit visuellement correct, mais il doit assurer une représentation précise de la structure sous-jacente. Par exemple, si deux arêtes se croisent, l'intersection doit faire partie du modèle et de ce fait vient couper les arêtes en deux. Il en résulte que les arêtes sécantes sont en réalité interdites dans le modèle.

Pour étudier ce modèle dans un cas pratique, nous avons choisi une application en géologie : la création de chenaux (Figure 1). La création de chenaux est un cas d'étude riche tout en étant simple car elle a recours à seulement deux phénomènes classiques en géologie : la sédimentation et l'érosion. Ces deux phénomènes entraînent l'apparition de nombreuses modifications topologiques. Par exemple, sur la figure, le début d'une érosion est représentée. Si elle se pour-

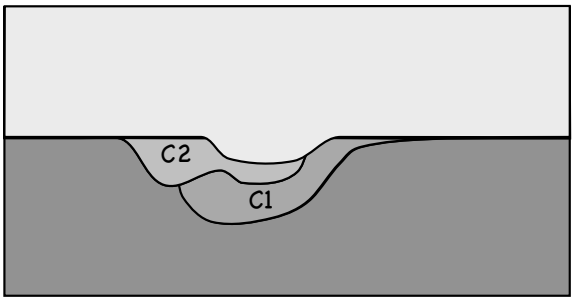


Figure 1: *Exemple de chenal obtenue par érosions et sédimentations successives.*

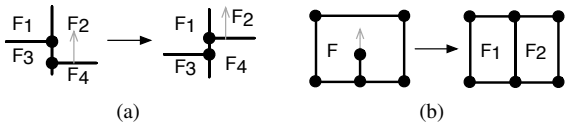


Figure 2: *Exemples de changements topologiques. (a) Glissement d'un sommet sur une autre avec changement des relations d'adjacences entre les quatre faces. (b) Scission d'une face F en F_1 et F_2 causée par le sommet en mouvement.*

suit, il est probable que les couches $C2$ et $C1$ finiront par être séparées, i.e. ne seront plus adjacentes. En outre, une forte érosion peut conduire $C1$ à être coupée en deux blocs $C1_1$ et $C1_2$ non adjacents. Sur le plan topologique, cette évolution peut se décrire par des événements : glissement et séparation d'interfaces(Figure 2a), fission de face(Figure 2b), etc.

Ce papier est organisé comme suit : la section 2, après avoir présenté les limites des systèmes actuels d'animation gérant la topologie, expose la démarche générale de conception de notre modèle d'animation. La section 3 décrit les diverses composantes de ce modèle. La section 4 illustre l'utilisation du modèle avec la création de chenaux en 2D. Enfin, la section 5 dresse le bilan de nos travaux et leurs perspectives.

2. Animation topologique

2.1. Travaux antérieurs

Plusieurs systèmes permettent de représenter des modèles structurés et dynamiques. Les L-systèmes [Lin68] [PL90], les map-L-systèmes [LR79], les systèmes vertex-vertex [Smi06], MGS [GM01] et la modélisation de la croissance interne du bois [GTM*05] en sont des exemples. Un L-système est une grammaire formelle utilisée afin de modéliser des processus de développement et de prolifération de plantes et de bactéries. Cette grammaire formelle comprend un alphabet V, un ensemble de constantes S, un axiome de départ ω et un ensemble de règles P d'évolutions du système. Les variantes de ce modèle portent principalement sur l'application des règles (utilisation du contexte, de conditions, probabilités). Les map-L-systèmes appliquent le principe des L-systèmes sur des graphes. Les systèmes vertex-vertex sont décrits par un graphe non orienté où les sommets représentent les sommets de la structure, et les liens représentent une permutation des arêtes autour des sommets. Le système évolue par un ensemble de modifications appliquées au cours du temps. Les travaux sur la modélisation de la croissance interne du bois utilisent un modèle topologique évoluant au cours du temps, comparable aux L-systèmes classiques mais en travaillant sur des volumes au lieu d'arêtes. La croissance est guidée par l'application séquentielle de règles choisies suivant le contexte. Comme pour les L-systèmes, les transformations consistent essentiellement en des subdivisions hiérarchiques d'un maillage. L'ensemble de ses approches reposent sur la théorie des langages, et permettent de faire évoluer des systèmes pouvant se réduire à des structures linéaires. MGS est un langage de programmation pour la transformation de structures et est essentiellement basé sur un système à base de règles de transformations. Toutes ces structures n'apportent pas réellement de notions de cohérence géométrique et topologique.

2.2. Démarche

Le modèle proposé ne repose pas sur une structure linéaire. Au contraire, il consiste en une animation d'une subdivision de l'espace. Il se base sur un modèle topologique, les cartes généralisées, que l'on exploite dans une structure temporelle. Plus précisément, le modèle représente l'animation comme une succession de cartes généralisées, où chaque carte représente un ensemble de modifications topologiques simultanées et supposées instantanées. Pour définir les instants où une carte est nécessaire et plus généralement, garantir que le modèle topologique reste cohérent lors des mouvements de ses entités, une gestion événementielle des modifications topologiques est adjointe au modèle. Un scénario fourni par l'utilisateur spécifie les mouvements globaux de la structure. Ce scénario est traduit en série d'événements et son interprétation aboutit à la génération de l'animation. Un modèle sémantique complète notre modèle d'animation en lui fournissant un mécanisme de désignation des entités, sur lequel le scénario peut s'appuyer et qui lui permet d'être exprimé en termes « haut-niveau », *i.e.* facilement compréhensibles par l'utilisateur final. La désignation est hiérarchique afin de renseigner le modèle sur l'origine des entités. En outre, une description séquentielle des transformations que subit la carte initiale est également générée lors de l'élaboration de l'animation, et s'appuie en grande partie sur le mécanisme de désignation. Cette description séquentielle indique, sous une forme compréhensible, l'ensemble des modifications locales subies par la structure et autorise ainsi une analyse ultérieure des phénomènes reproduits.

3. Modèle événementiel d'animation topologique

Comme nous l'avons mis en évidence, notre modèle d'animation topologique est composé de trois parties principales : le modèle structurel, le modèle événementiel et le modèle sémantique. Le modèle structurel porte l'information topologique et géométrique, en incluant une dimension temporelle. Le modèle événementiel vise à détecter et contrôler les modifications topologiques et vient modifier le modèle structurel. Le modèle sémantique associe un nom à chaque arête pour désigner les différentes cellules de la subdivision de manière hiérarchique. Nous détaillons ces trois modèles dans cette partie.

3.1. Modèle structurel

Pour une représentation précise des voisinages, le modèle structurel s'appuie obligatoirement sur un modèle topologique. Il existe de nombreux modèles permettant de représenter de telles structures [Edm60] [May67] [Wei88] [Lie94] [LFB07]. Nous avons choisi les cartes généralisées (n-g-cartes) [Lie94], car les n-g-cartes sont définies de manière homogène en toutes dimensions, ce qui simplifie la définition du modèle et des opérations. Nous rappelons les grands principes de ces structures ci-dessous. Nous exploitons ces structures dans une structure séquentielle que nous décrivons ensuite.

3.1.1. Structure topologique : n-G-Cartes

Les g-cartes [Lie94] représentent des objets par leurs bords (B-Rep). Elles modélisent les quasi-variétés cellulaires orientées ou non, avec ou sans bord. Les objets géométriques sont subdivisés en cellules (sommets, arêtes, faces, etc.) reliées entre elles par des relations d'adjacences/incidences (Figure 3).

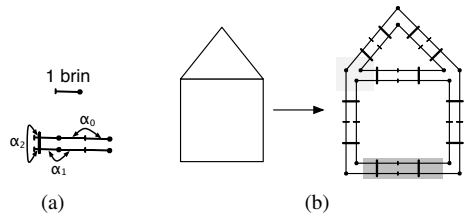


Figure 3: *(a) Convention utilisée pour la représentation d'un brin et des liaisons. (b) Exemple de deux faces collées entre elles et leur représentation par une 2-g-carte fermée. En gris clair : un exemple d'orbite sommet, en gris foncé : un exemple d'orbite arête.*

Définition 1 Une g-carte de dimension n, ou n-g-carte est un $(n+2)$-uplet $G = (B, \alpha_0, \ldots, \alpha_n)$ tel que :
– B est un ensemble fini de brins ;
– $\alpha_0 \ldots \alpha_n$ sont des involution sur B ;
– $\alpha_i\alpha_j$[†] est une involution pour $0 \leq i < i+2 \leq j \leq n$.

Définition 2 L'orbite $< \Phi > (b)$, pour un ensemble de permutations Φ, est l'ensemble des brins de B que l'on peut atteindre à partir de b par une composition quelconque des permutations de Φ. Si Φ est égal à l'ensemble de toutes les involutions $\alpha_0, \ldots, \alpha_n$ alors $< \Phi > (b)$ est une composante connexe de G incidente au brin b.

Définition 3 Une n-G-Carte $G = (B, \alpha_0, \ldots, \alpha_n)$ est fermée ssi $\forall i \in \{0, \ldots, n\}, b\alpha_i \neq b$.

Dans une n-g-carte, chaque i-cellule (cellule de dimension i) est obtenue par une *orbite* $< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n >$.

Les orbites permettent principalement de parcourir la n-g-carte et d'associer des informations aux différentes i-cellules.

3.1.2. Modèle temporel : keyframe

Le but de ce modèle est de représenter l'animation d'objets structurés comme une succession de modifications topologiques (Figure 4). Nous prenons comme hypothèse qu'une modification topologique est instantanée mais que plusieurs modifications peuvent être simultanées. Notre approche s'inspire largement de la technique d'animation par images-clefs. Cependant, ici, une nouvelle image, (ou plutôt instant-clef) n'est introduite dans le modèle que lorsque la structure subit une ou des modifications topologiques. Pour cela, à l'instant-clef correspond en réalité une nouvelle carte. Entre deux instants-clefs consécutifs, il n'y a pas de changement topologique, seul le plongement change.

Plus précisément, notre modèle est une succession temporelle de 2-g-cartes fermées connexes, où chaque g-carte est créée à partir de la précédente. En pratique, à un instant i_{k+1}, la dernière carte de l'instant i_k est dupliquée, sa date est changée pour correspondre à i_{k+1}. Ensuite, cette carte est altérée par toutes les modifications topologiques se produisant à cet instant. Cette méthode de construction implique d'avoir un ensemble de modifications topologiques triées selon la date [LSM06].

La structure présentée ne décrit que les changements topologiques apparus au cours du temps. Il est nécessaire d'adjoindre aux cartes un plongement temporel capable de décrire les trajectoires des entités jusqu'à la prochaine modification topologique. Pour cela, nous avons choisi de ne considérer que le plongement le plus simple, à savoir celui des sommets. À chaque 0-cellule est associée une fonction $f : t \rightarrow \mathbb{R}^2$. Les objets sont ici plongés dans $\mathbb{R}^2$ mais ils pourraient aussi bien être plongés dans $\mathbb{R}^3$ pour modéliser

[†] $\alpha_i\alpha_j$ est la notation qui correspond à la composition $\alpha_j \circ \alpha_i$. $b\alpha_i\alpha_j$ correspond à l'application de cette composition à un élément b de B.

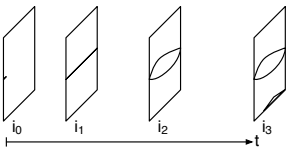


Figure 4: *Modèle temporel = succession de 2-g-cartes ordonnées selon le temps t.*

l'évolution de surfaces dans l'espace. En effet, la restriction de la dimension à 2 ne concerne que la topologie.

À noter que notre plongement est une fonction temporelle à valeur dans l'espace et non une position dans l'espace-temps. Cependant, notre modèle est équivalent à un modèle spatio-temporel (i.e. en dimension 2D+t). En effet, notre modèle peut se construire à partir d'un modèle espace-temps en créant des coupes temporelles, i.e. des intersections avec un hyperplan perpendiculaire à l'axe du temps. Tant que la topologie des différentes intersections successives d'un hyperplan avec la structure ne change pas, nous nous trouvons entre deux instants-clefs où seuls les plongements géométriques changent en fonction de t. Inversement, le passage de notre modèle vers un modèle spatio-temporel s'effectue d'abord par extrusion des 2-g-cartes le long de l'axe temporel, puis par l'union des hyper-volumes ainsi fabriqués et disposés l'un après l'autre le long de l'axe temporel (Figure 5).

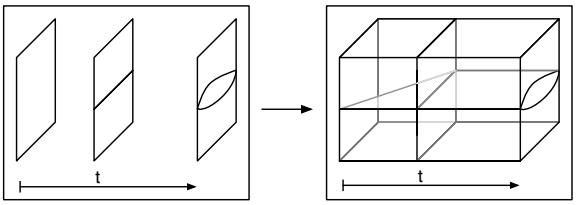


Figure 5: *Passage du modèle temporel à un modèle spatio-temporel par l'union des extrusions des 2-g-cartes suivant le plongement des sommets.*

3.2. Modèle événementiel

Le modèle structurel suffit à lui seul à décrire une animation par une suite d'opérations topologiques et d'affectations de plongements sommets. Le problème est de déterminer *a)* à quel instant a lieu chaque modification topologique, *b)* les opérations topologiques à appliquer pour mettre en œuvre les modifications et *c)* les nouveaux plongements.

Pour répondre à la question *a)*, un système à événements discret est utilisé. Une approche prédictive permet de déterminer, en fonction des évolutions de la géométrie, la date d'un événement topologique, c'est-à-dire l'instant où une entité rencontre une autre entité. Les événements que nous gérons peuvent provenir de diverses origines. Certains sont générés directement par le scénario, i.e. aux évolutions haut-niveau du scénario peuvent correspondre des événements initiaux. D'autres événements résultent de l'évolution topologique du système. Par exemple, si un point se déplace le long d'une arête, un événement est déclenché lorsque le point atteint l'extrémité de cette arête. Enfin, certains événements résultent de la collision entre des entités initialement indépendantes. Pour prédire l'instant où ces événements ont lieu, un système de détection de collision continue est utilisé. Il est issu d'une application en 2D des équations de Provot [Pro97]. Signalons que, pour mener les calculs à leur terme, le système prédictif fait l'hypothèse de mouvements rectilignes uniformes des sommets. À partir de cette détection, un événement est alors complètement défini par sa date et l'ensemble des entités topologiques impliquées.

La gestion par événements discrets est similaire à celle utilisée habituellement en animation [DZ93]. Une fois détectés, les événements sont ajoutés dans une file à priorités triée suivant le temps. Dans l'ordre de leur apparition, les événements sont extraits de la file puis traités. Le traitement d'un événement débute par la vérification de la validité de l'événement (en effet, des modifications topologiques précédentes ont pu rendre caduque l'apparition d'un événement précédemment détecté [DZ93]). Si l'événement est valide, il est traité suivant le contexte de la scène et entraîne une série de transformations topologiques, géométriques et de désignations (voir le modèle sémantique). En effet, les modifications à appliquer dépendent de l'application et généralement du phénomène simulé.Ces transformations sont appliquées soit à la 2-g-carte courante si la date qui lui est associée est égale à la date de l'événement, soit à une nouvelle 2-g-carte créée par duplication de la précédente et dont la date est celle de l'événement. Cependant, pour répondre de façon complète aux questions *b)* et *c)*, nous étudions la possibilité d'utiliser un cadre formel à base de règles pour cette étape (voir [PCLG*07, GM01]).

3.3. Modèle sémantique

Le modèle sémantique vise deux objectifs : représenter de façon explicite les modifications subies par la structure au cours du temps et fournir une représentation de l'historique des entités topologiques. Pour remplir ces deux objectifs, nous utilisons un mécanisme de désignation des entités. La désignation consiste à associer des noms aux cellules de la scène et ces noms sont utilisés comme paramètres des descriptions et des opérations topologiques et géométriques. Ce mécanisme forme ainsi un pont entre l'aspect « haut-niveau »(descriptif) de notre modèle, dans lequel l'utilisateur contrôle l'animation, et l'aspect « bas-niveau »(structurel) sur lequel s'applique les décisions de l'utilisateur. Cette désignation est hiérarchique. Par exemple, l'opération d'insertion de n sommets sur une arête résulte en $n+1$ arêtes dont les désignations sont préfixées par la désignation de l'arête d'origine.

En 2D, seules les 1-cellules (arêtes) sont désignées car, à partir des arêtes, il est facile de désigner soit un sommet, soit une face. En effet, le modèle des g-cartes est une représentation par les bords, ce qui se traduit en 2D par la propriété suivante : une arête est incidente à au plus deux faces et deux sommets. Donc chaque arête est associée à un identifiant qui permet de la désigner (Figure 6). Notons au passage que comme la désignation s'opère à partir d'une arête, il existe plusieurs moyens de désigner les sommets et les faces (le nombre de façons de les désigner est respectivement égal au degré du sommet et au nombre d'arêtes formant le bord de la face).

Concrètement, le nom d'une arête est affecté à l'un des quatre brins la constituant (Figure 6). La désignation des 0-cellules et des 2-cellules s'effectue à partir de ce brin. Ainsi, un sommet est désigné par le nom d'une arête et sa position sur l'arête (*begin*, *notbegin*). Une arête est désignée par son nom. Une face est désignée par le nom d'une arête et sa position par rapport à l'orientation de l'arête : $left$ ou $notleft$.

Enfin, lors des traitements des événements, le modèle sémantique génère la liste des opérations topologiques appliquées aux entités désignées qui ont été altérées. Cette liste forme une liste séquentielle de modifications, en d'autres termes, un script, que la carte initiale subit au cours du temps. Ce script retrace l'historique de construction du modèle et permet d'analyser, a posteriori, le résultat de l'animation.

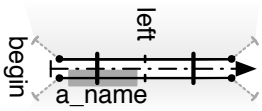


Figure 6: *Mécanisme de désignation des 0,1,2-cellules à partir du brin portant la désignation de la 1-cellule.*

4. Application à la géologie : création de chenaux

Après avoir décrit notre modèle d'animation, nous montrons maintenant son application dans l'étude de deux phénomènes géologiques classiques, la sédimentation et l'érosion, qui interviennent dans la construction de chenaux. Pour cela, nous définissons un modèle de sédimentation et un modèle d'érosion ainsi que les paramètres pour les manipuler. Enfin, nous décrivons comment les intégrer dans le modèle d'animation.

4.1. Modèle de sédimentation

La sédimentation est l'ensemble des processus par lesquels des particules en suspension se déposent. Selon les principes de stratigraphie énoncés par Stenon (1669), les sédiments se déposent en couche à peu près horizontales.

Nous considérons un modèle de sédimentation suivant une déposition strictement horizontale. Cette sédimentation débute par le remplissage des zones les plus basses. Ensuite, le niveau monte et si deux couches issues d'une même sédimentation se rencontrent, elles fusionnent (Figure 7).

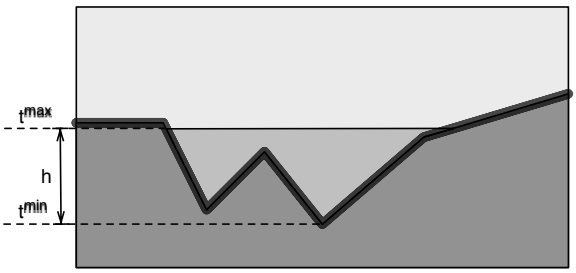


Figure 7: *Paramètres du modèle de sédimentation.*

La sédimentation est définie par cinq paramètres : un préfixe pour désigner les différents morceaux créés par la sédimentation, une zone de sédimentation délimitée par deux arêtes, une date de début *t_begin*, une date de fin *t_end* et une hauteur *h*. Ce modèle simplifié considère que la vitesse de sédimentation est constante (ie. : $v = \frac{h}{t\_end - t\_begin}$).

Le traitement de ce phénomène s'effectue en deux étapes :

1. recherche des minima locaux sur la zone de sédimentation (Figure 8a) ;
2. ajout des événements de créations d'interfaces aux temps calculés pour chaque minimum (i.e. instant de l'arrivée du niveau de sédimentation à la hauteur du minimum).

L'événement de création d'interface insère une arête dans la face à sédimenter au niveau du minimum courant (Figure 8b). Les extrémités de l'arête sont plongées par une

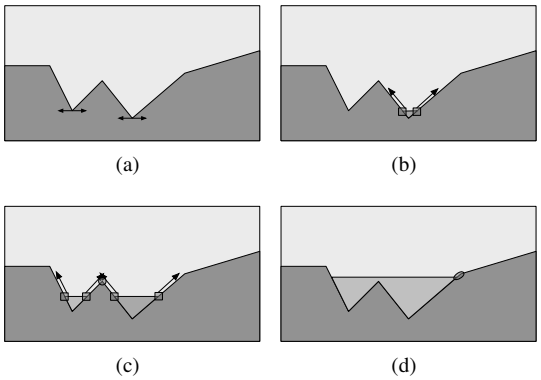


Figure 8: *Étapes d'une sédimentation. (a) Recherche des minima locaux. (b) Création de la première face et glissement de ses extrémités. (c) Création d'une nouvelle face et glissement de ses extrémités. (d) Fusion des deux blocs avec glissement des extrémités.*

fonction d'interpolation des coordonnées du minimum courant et du prochain point par rapport à la vitesse de sédimentation. Deux nouveaux événements de collisions sont générés s'il est prévu que les extrémités des arêtes touchent les prochains sommets dans le futur.

L'événement généré peut être traité de deux façons suivant le contexte local. Soit l'arête suivante est issue de la sédimentation courante (Figure 8c), soit elle ne l'est pas (Figure 8d). Dans le premier cas, les deux arêtes sont désidentifiées du bord courant et sont fusionnées. Il en résulte une fusion des faces. Dans le second cas, le sommet mobile passe au delà du sommet qu'il touche puis glisse sur l'arête suivante (modification du plongement). La collision avec l'autre extrémité de l'arête est ajoutée à la file des événements. Ce processus se répète jusqu'à la fin du phénomène de sédimentation.

4.2. Modèle d'érosion

L'érosion est l'ensemble des processus de dégradation et de transformation du relief.

Pour simplifier le paramétrage de ce phénomène, le profil de l'érosion est en forme de cuve. La forme n'a ici pas d'incidence sur le traitement. L'érosion débute par la déformation de la surface délimitée par deux arêtes. Si la surface d'érosion touche une autre surface *s*, celle-ci est modifiée en suivant le profil de la surface d'érosion (Figure 9).

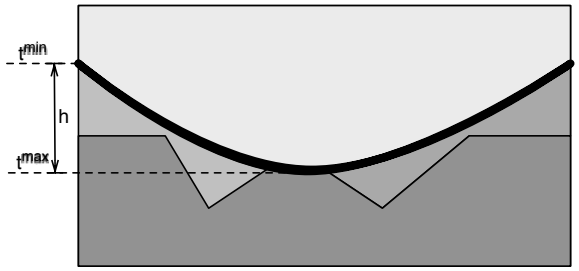


Figure 9: *Paramètres du modèle d'érosion.*

L'érosion est définie par quatre paramètres : la zone d'érosion délimitée par deux arêtes, une date de début *t_begin*, une date de fin *t_end* et une hauteur *h*.

Le traitement de ce phénomène s'effectue en trois étapes :

1. rééchantillonnage de la surface d'érosion (Figure 10a) ;
2. calcul et mise à jour des plongements des sommets définissant la surface d'érosion (Figure 10a) ;
3. recherche de la première collision entre la surface d'érosion et le reste du modèle, puis création de l'événement correspondant (Figure 10b).

L'événement de collision (Figure 10c) dépend du contexte : soit la surface d'érosion est "prioritaire" et érode les surfaces qu'elle touche, soit elle n'est pas "prioritaire"

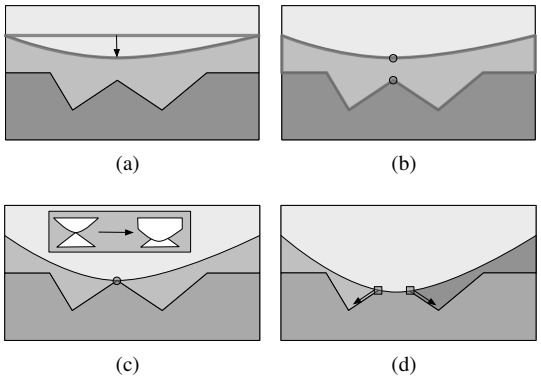


Figure 10: *Étapes d'une érosion. (a) Rééchantillonnage de la surface d'érosion et changement du plongement des sommets. (b) Recherche de la date la plus proche de collision. (c) À l'instant de la collision, traitement de l'événement suivant le contexte, ici la surface d'érosion est prioritaire sur le reste. (d) La surface d'érosion a érodé le pic : une nouvelle face est créée et les points de contacts glissent.*

et elle est scindée en plusieurs morceaux lors de la collision. Cette notion de priorité est explicitée dans [Per98] et adaptée à la modélisation informatiques de structures géologiques dans [BPRS01]. Dans notre exemple, une fois traités, les points de contacts de la surface érodée glissent le long de la surface d'érosion. Une nouvelle recherche de la plus proche intersection dans les deux nouvelles faces est réalisée. Les points qui glissent génèrent également des événements de collisions entre le point de contact et les sommets de la surface d'érosion (Figure 10d).

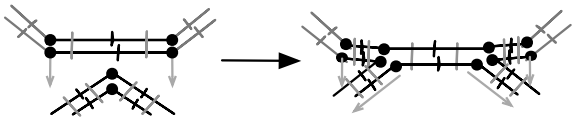


Figure 11: *Traitement de la collision entre une arête d'érosion et un sommet du modèle au niveau topologique et géométrique. Les flèches correspondent aux trajectoires des sommets.*

La figure 11 illustre un exemple de traitement de collision entre une arête appartement à la surface d'érosion *a* et un sommet du modèle *s*. Deux nouveaux sommets sont insérés sur l'arête *a*, les arêtes incidentes au sommet *s* sont déliées, et liées aux sommets précédemment insérés. Il faut ensuite affecter un plongement aux nouveaux sommets, qui va correspondre au point d'intersection de l'arête *a* toujours en mouvement et des arêtes qui étaient liées à *s*.

4.3. Résultats

L'écriture de scénarii d'animation passe par l'intermédiaire de script. Voici une ébauche de langage adaptée à la géologie permettant de construire le chenal simple de la figure 12 :

```
a = creerAnimation(8, 6, 0);
a.ajouter(creerChenal(0, "profil_7.v", "C1"));
a.ajouter(creerSedimentation(10, 20, 3, "Sed01",
      "border_left_0", left,
      "border_right_1", left);
a.ajouter(creerErosion(30, 40, 3,
      "Sed01_0", begin, "Sed01_0"));
```

Ce script crée une nouvelle animation de taille 8x6, définit la première image-clef à la date 0 et positionne le profil de chenal $C1$(Figure 12a). Ensuite, il ajoute une période de sédimentation de la date 10 à 20 pour une hauteur totale de 3. Cette sédimentation affecte toute la scène (du bord gauche au bord droit) et crée des interfaces disjointes dont la désignation est préfixée par *Sed*01. Ce phénomène a pour effet de créer deux nouvelles interfaces *Sed*01_0 et *Sed*01_1 dont les extrémités glissent le long du bord de $C1$ (Figure 12b). Au niveau du pic, *Sed*01_0 et *Sed*01_1 se rencontre et fusionnent en *Sed*01_0 (Figure 12c). Enfin, le script ajoute une période d'érosion de la date 30 à 40 pour une hauteur totale de 3 appliquée à l'interface *Sed*01_0. Ce phénomène a pour effet d'échantillonner la surface d'érosion en affectant des trajectoires aux nouveaux sommets (Figure 12d). Le moteur de collision prédit une collision entre la surface d'érosion et le pic de $C1$. La collision est traitée en donnant la priorité à la surface d'érosion (Figure 12e).

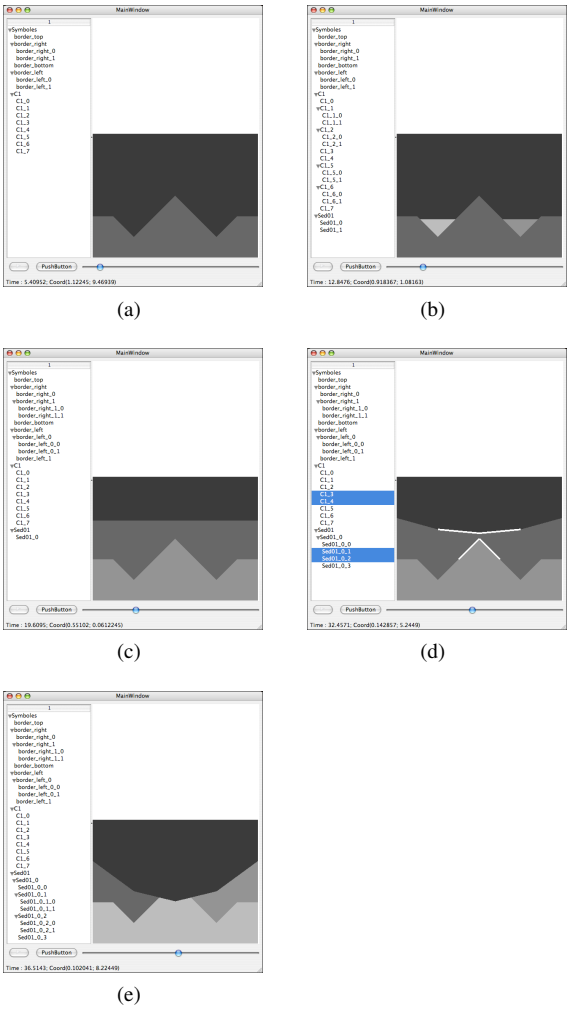


Figure 12: *(a) Création d'un chenal à partir d'un profil. (b) Début de sédimentation créant deux blocs disjoints. (c) Fusion des deux blocs. (d) Début de l'érosion et détection de la prochaine collision causée par le phénomène. (e) Résultat après le traitement de la collision entre la surface d'érosion et le pic.*

5. Conclusion

Nous proposons un modèle d'animation 2D basé sur l'évolution dynamique de structures topologiques. Ce modèle est composé d'une suite de 2-g-cartes correspondante à un ensemble de modifications topologiques instantanées. Chaque 2-g-carte possède un système de désignation et de plongement temporel. La description de l'animation est réalisée par des actions entraînant une suite d'événements topologiques. La prévision de collisions détermine les événements de changements topologiques. Chaque événement est traité suivant son contexte local et suivant l'application. Pour illustrer ce modèle, nous avons utilisé le cas de la création de chenaux au travers de la description de deux phénomènes, la sédimentation et l'érosion. Ces phénomènes ont été décomposés en événements locaux de créations et de collisions.

Nous travaillons actuellement sur l'amélioration de nos modèles de sédimentation et d'érosion pour que la construction de chenaux puisse être entièrement contrôlée par le géologue. Puis, en suivant la même méthodologie, nous allons étudier d'autres phénomènes géologiques, comme la création de failles et les glissements de couches du sous-sol : d'une part pour enrichir les possibilités de modélisation de notre application ; et d'autre part pour compléter notre modèle d'animation en intégrant des situations que nous n'avons pas eu l'occasion de rencontrer avec la sédimentation et l'érosion.

Nous souhaitons ensuite formaliser le traitement des événements en nous orientant vers un système de reconnaissance de motifs et de règles de réécritures de graphes [PCLG*07]. Cette approche devrait nous permettre de déterminer plus simplement les actions à réaliser selon le contexte local de l'événement, renforçant ainsi l'indépendance de notre modèle d'animation par rapport à des applications particulières.

Nos objectifs à plus long terme sont nombreux : d'abord,

tester et enrichir notre modèle face à de nouveaux contextes d'utilisation en 2D. Puis, étendre le modèle à la 3D : la structuration en modèles structurel, événementiel et sémantique ne changera pas mais nous devrons les enrichir pour contrôler les nouveaux cas d'évolutions des structures topologiques qui apparaîtront avec le changement de dimension (par exemple, l'intersection de l'intérieur d'une face avec un sommet). Enfin, l'édition interactive des scénarii pour modifier certains paramètres et rejouer l'évolution du modèle nous intéresse particulièrement. Nous comptons suivre l'approche de la « nomination » [BAMSB05] pour résoudre ce problème.

References

[BAMSB05] BABA-ALI M., MARCHEIX D., SKAPIN X., BERTRAND Y. : Intégration des opérations de nomination dans un modèle géométrique 3d. In *Journées de l'Association Française d'Informatique Graphique* (November 2005).

[BPRS01] BRANDEL S., PERRIN M., RAINAUD J.-F., SCHNEIDER S. : Geological interpretation makes earth models easier to build. In *EAGE 63rd Conference, Extended Abstacts* (June 2001).

[DZ93] DWORKIN P., ZELTER D. : A new model for efficient dynamic simulation. In *EUROGRAPHICS Workshop on Computer Animation and Simulation (EGCAS)* (Barcelone, Sept. 1993), pp. 135–147.

[Edm60] EDMONDS J. : A combinatorial representation for polyhedral surfaces. In *Notices*, vol. 7. Amer. Math. Soc., 1960.

[GM01] GIAVITTO J.-L., MICHEL O. : Mgs : a rule-based programming language for complex objects and collections. *Electr. Notes Theor. Comput. Sci. 59*, 4 (2001).

[GTM*05] GUIMBERTEAU G., TERRAZ O., MÉRILLOU S., GHAZANFARPOUR D., PLEMENOS D. : *Internal wood growth simulation based on subdivised 3D objects*. Tech. rep., Laboratoire MSI, 2005.

[LFB07] LIENHARDT P., FUCHS L., BERTRAND Y. : *Informatique graphique, modélisation géométrique et animation.*, vol. 1 of *Traitement du Signal et de l'Image*. Hermès, 2007, ch. Modèles topologiques, pp. 49–93. sous la direction de D. Bechmann et B. Péroche.

[Lie94] LIENHARDT P. : n-dimensional generalised combinatorial maps and cellular quasimanifolds. *International Journal of Computational Geometry and Applications* (1994).

[Lin68] LINDENMAYER A. : Mathematical models for cellular interactions in development. *Journal of Theoretical Biology 18* (1968), 280–315.

[LR79] LINDENMAYER A., ROZENBERG G. : Parallel generation of maps : Developmental systems for cell layers. In *Proceedings of the International Workshop on Graph-Grammars and Their Application to Computer Science and Biology* (London, UK, 1979), Springer-Verlag, pp. 301–316.

[LSM06] LÉON P., SKAPIN X., MESEURE P. : Topologically-based animation for describing geological evolution. In *International Conference on Computer Vision and Graphics* (September 2006).

[May67] MAY J.-P. : *Simplicial Objects in Algebric Topology*, première ed., vol. 11 of *Van Nostrand Mathematical Studies*. Van Nostrand, 1967.

[PCLG*07] POUDRET M., COMET J.-P., LE GALL P., ARNOULD A., MESEURE P. : Topology-based geometric modelling for biological cellular processes. In *1st International conference on Langage and Automata Theory and Applications (LATA 2007)* (Tarragona, Spain, Mars 2007). à paraître.

[Per98] PERRIN M. : Geological consistency : an opportunity for safe surface assembly and quick model exploration. 4–5.

[PL90] PRUSINKIEWICZ P., LINDENMAYER A. : *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. Springer, October 1990.

[Pro97] PROVOT X. : Collision and self-collision handling in cloth model dedicated to design garments. *Graphics Interface* (1997), 177–189.

[Smi06] SMITH C. : *On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling*. PhD thesis, University of Calgary, April 2006.

[Wei88] WEILER K. : The radial edge structure : A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications : Selected and Expanded Papers from the Ifip Wg 5.2 Working Conference* (1988), Wozny M.-J., McLaughlin H.-W., Encarnacão J.-L., (Eds.), Elsevier Science, pp. 3–36.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Texturation d'environnements urbains par système mobile avec un couplage Caméra/Télémètre Laser

Jean-Emmanuel Deschaud [1], Xavier Brun [1] et François Goulette [1]

[1]Ecole des Mines de Paris
60, boulevard Saint Michel
75272 PARIS Cedex 06, FRANCE
xavier.brun@ensmp.fr, jean-emmanuel.deschaud@ensmp.fr, francois.goulette@ensmp.fr
http://caor.ensmp.fr/

Abstract

Nous présentons dans cet article un nouveau système temps réel de texturation d'environnements urbains et routiers monté sur un véhicule. Le système d'acquisition est basé sur deux capteurs, un télémètre laser et une caméra avec un objectif fish-eye. Nous produisons des points 3D colorés ou des modèles surfaciques triangulés et texturés de l'environnement traversé par le véhicule. Nous expliquons nos choix de l'objectif fish-eye pour sa couverture large du monde environnant la voiture (180°*). Une partie importante du processus consiste à calibrer le système : nous avons besoin de récupérer la transformation rigide entre le repère laser et le repère de la caméra dont le modèle de projection est différent du modèle usuel "pin-hole". Nous présentons une nouvelle méthode plus rapide pour déterminer les paramètres de projection du fish-eye. Une fois que notre système a été calibré, l'acquisition et le traitement des données se fait "en vol". Des résultats de rues modélisées illustrant les possibilités de ce nouveau système de scanner sont également présents dans cet article.*

We present a new Mobile Mapping System mounted on a vehicle to reconstruct outdoor environment in real time. The scanning system is based on two sensors, a laser range finder and a camera equipped with a wide-angle fisheye lens. We can produce 3D coloured points or 3D textured triangulated models of the nearby environment of the vehicle. We explain our choice of the fisheye lens for its large aperture angle covering a large part of the world surrounding the car (180°*). An important part of the process is to calibrate the system: we need to get the rigid transformation between the two frames, and the fisheye projection model which is different from the usual "pin-hole" model. We present a new and faster way to determine the fisheye projection parameters. Once the system has been calibrated, the data acquisition and processing to create models are done "on-the-way". Results illustrating in city streets the possibility of the new scanning system are presented.*

1. Introduction

Les modèles d'environnements urbains et routiers sont utiles dans de nombreuses applications comme l'architecture, la cartographie 3D, la navigation terrestre, le tourisme virtuel, la réalité virtuelle, les jeux vidéos, etc... Pour une modélisation réaliste, une numérisation précise de la géométrie 3D ainsi qu'une bonne texturation de l'environnement à l'échelle désirée (par exemple, le sol pour une application de navigation terrestre) sont nécessaires. Cependant, étant donné un niveau de détail déterminé ainsi qu'une taille et une surface choisie de l'environnement, nous avons affaire à un problème de quantité d'information à acquérir, à traiter et à conserver. Pour obtenir la géométrie 3D d'un environnement extérieur au niveau du sol, on peut utiliser des méthodes passives comme la photogrammétrie ou la stéréovision pour calculer la localisation de points spécifiques de la scène, ou alors utiliser des méthodes actives comme l'utilisation d'un laser pour récupérer la structure de la surface scannée. Les méthodes actives donnent généralement des résultats plus denses. Plusieurs laboratoires ont étudié

l'utilisation d'un télémètre laser sur un véhicule dans le but de scanner de grandes zones de l'environnement. Une possibilité est de définir un certains nombre de zones d'intérêts à scanner de manière statique et de faire la fusion des nuages de points 3D des différentes localisations. Cela requiert bien sûr un traitement supplémentaire ([ASG*01]; [SA00]). Une autre possibilité est d'utiliser le véhicule en tant que système de scanner, son mouvement définissant une des directions de l'environnement à modéliser ([FZ03]; [ZS03]). Dans la méthode citée précédemment, le traitement des données collectées durant l'acquisition est faite à part et nécessite plusieurs heures de calculs ce qui ne permet pas d'avoir un système entièrement embarqué.

Nous avons développé un système de télémètre laser embarqué, qui a été conçu spécialement pour l'acquisition de la géométrie 3D ainsi que la texture des villes et des routes. Tout ceci se fait à la vitesse du véhicule. Notre système est une plateforme composée d'un véhicule équipé de capteurs de localisation (GPS, centrale inertielle) et de capteurs de perception (télémètre laser, caméra). Le télémètre laser, placé à l'arrière du véhicule, scanne des plans verticaux perpendiculaires à la direction du véhicule. Durant le mouvement de la voiture, à vitesse basse ou normale, le laser va scanner tout l'environnement traversé. La géométrie et la texture sont capturés ensemble par une caméra CCD avec objectif fish-eye et le télémètre laser. Combiné à une information de géo-référencement, nous sommes capable de créer des points 3D seuls, des points 3D colorés, des modèles à facette seuls ou texturés de la géométrie de l'environnement. Pour le traitement des données des capteurs, nous utilisons un logiciel temps réel embarqué spécialement étudié pour notre plateforme.

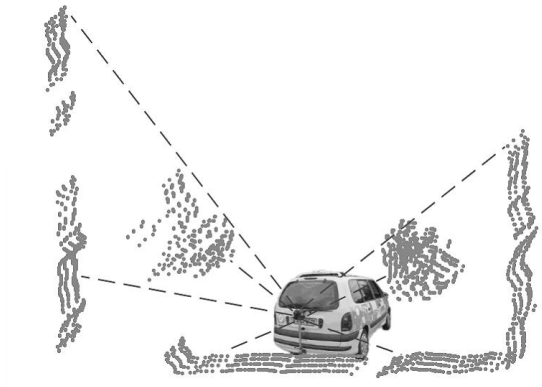


Figure 1: *LARA-3D, prototype du système avec télémètre laser embarqué*

Nous présentons dans ce papier le système complet, la méthodologie et les algorithmes développés. Nous discutons tous les choix qui ont été faits. La section 2 présente la plateforme expérimentale LARA-3D; la section 3 est dédiée à la description de reconstruction de la géométrie; enfin, la section 4 détaille les algorithmes utilisés pour ajouter la couleur et les textures.

2. La plateforme LARA-3D

2.1. Analyse fonctionnelle

Dans le but de produire les différents résultats avec les points 3D seuls, les points colorés, les modèles à facettes seuls ou texturés à partir des informations du télémètre laser, de la caméra et des capteurs de localisation (GPS et centrale inertielle), une analyse fonctionnelle nous amène à définir plusieurs fonctions réalisées par le système embarqué :

- Localisation: besoin de la position, de la vitesse et de l'orientation précis en temps réel et à haute fréquence du véhicule;
- Géo-référencement: transformer les données distances du laser en points 3D exprimés dans un repère commun géo-référencé;
- Triangulation: production de facettes triangulées à partir des points 3D;
- Colorisation: trouver la correspondance entre les images de la caméra et les points 3D;
- Texturation: ajout des textures (extraits des images de la caméra) aux facettes triangulées.

2.2. Matériel et logiciel

Nous avons implémenté les fonctions décrites sur un véhicule prototype appelé LARA-3D. Il s'agit d'une Renault Espace équipée de capteurs de localisation (GPS, centrale inertielle), de capteurs de perception (télémètre laser rotatif, caméras) et d'un ordinateur embarqué avec un environnement logiciel adapté et une grande capacité de stockage.

Pour la localisation, nous utilisons une antenne GPS AgGPS132 de Trimble (AgGPS 2000), délivrant des positions absolues différentielles à une fréquence de 10 Hz avec un temps de latence compris entre 30 et 150ms. La centrale Inertielle est une VG600CA-200 de Crossbow monté en strapdown, délivrant des données à une fréquence de 84 Hz. Le télémètre laser est une LD Automotive d'IBEO, délivrant 1080 distances par rotation à une fréquence de 10 rotations par seconde. Sur notre plateforme, nous utilisons un bi-processeur Pentium III à 750 MHz avec Windows 2000.

Dans le but d'ajouter de la couleur et des textures, nous avons considéré de nombreuses possibilités mais nous avons finalement choisi d'utiliser une caméra CCD avec un objectif fish-eye pour une capture très grand angle. Pour avoir une redondance photogrammétrique, nous avons choisi une caméra CCD matricielle pour obtenir la scène complète à chaque image (différent d'une caméra CCD linéaire). Sachant que l'acquisition des données doit être faite durant le mouvement du véhicule, nous avons besoin d'une

fréquence de plusieurs images par seconde. La résolution choisie est typique de ce genre de produit (Table 1).

Table 1: *Caractéristiques des éléments*

Télémètre laser	Nombre de points	1080 par profil
	Angle couvert	270°
	Fréquence profils	10 Hz
CCD couleur	Nombre de pixels	780x582 pixels
	Taille CCD (1/2"')	$6.4mm \times 4.8mm$
	Fréquence images	25 Hz
Objectif fish-eye	Focale	1.4 mm
	Angle d'ouverture	$185° \times 185°$

Le télémètre laser couvre un angle de 270°. Nous avons fait une comparaison des angles couverts par notre caméra avec trois objectifs différents. Les objectifs fish-eye sont des lentilles spécifiques utilisant des principes optiques différents de la normale, ce qui permet d'obtenir des longueurs de focales très courtes et ainsi de capturer la moitié de l'espace avec une seule caméra. Pour couvrir la même zone que le scanner, seuls deux caméras avec objectif fish-eye sont nécessaires, ce qui aurait nécessité au moins trois caméras avec un objectif classique. Cela explique notre choix d'un système composé d'une caméra CCD avec objectif fish-eye pour les images et d'un télémètre laser pour capturer la géométrie (Figure 2.2).

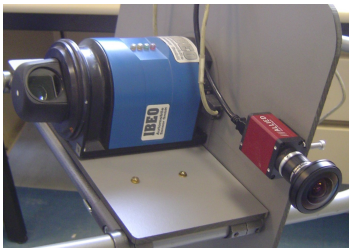


Figure 2: *Système de vision couplé*

Du point de vue logiciel, nous utilisons un environnement spécifique: une application temps réel embarquée nommée RTMAPS (Intempora). RTMAPS permet d'implémenter des algorithmes de traitement en C++ et de les tester sur des données réelles déjà acquises, c'est à dire de faire rejouer les données à des vitesses plus faibles ou plus grandes que lors de l'acquisition originelle. Cela permet d'avoir une détermination facile des temps de calcul. L'environnement permet aussi d'exécuter des algorithmes non optimisés, de mesurer les temps de calcul et de déterminer si il est possible d'accélérer le rejeu ou si il est nécessaire d'optimiser les algorithmes. Nous considérons qu'un algorithme supporte le temps réel quand le rejeu à vitesse normale permet le traitement de données sans perte de données ou de synchronisation. Dans ce cas, il est possible d'installer le logiciel sur le PC embarqué pour faire simultanément l'acquisition et le traitement. Les différentes fonctions décrites précédemment dans l'analyse fonctionnelle ont été implémentées en tant que composants séparés dans RTMAPS qui peuvent ensuite être liés entre eux dans une interface de flux de données.

3. La géométrie 3D

3.1. Localisation du véhicule

Pour la localisation précise temps réel, nous utilisons un couple GPS-Centrale Inertielle [ANL*03]). Le système de mesure inertielle donne seulement des informations dérivées (accélérations et vitesses de rotation) qui nécessitent d'être intégrées pour donner la position, l'orientation et la vitesse du véhicule (Inertial Navigation System, INS). Ces informations à haute fréquence sont précises mais soumises à des dérives à long terme. L'information GPS donne des informations absolues (position et vitesse), mais à une faible fréquence et pas de façon régulier. La combinaison des deux, par fusion de données, permet d'obtenir une position précise et absolue du véhicule ainsi que son orientation et sa vitesse.

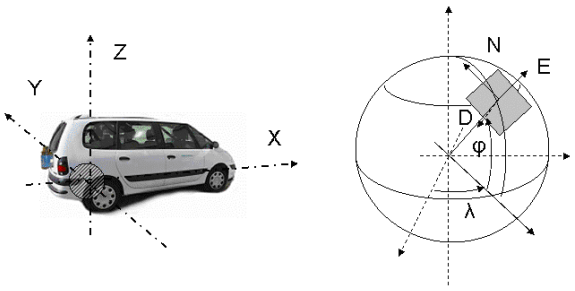


Figure 3: *Référentiels*

Dans le but d'obtenir des temps de calcul plus rapides, nous effectuons quelques simplifications dans les équations du système de navigation inertielle ([FB98]) (par exemple en négligeant le terme de Coriolis qui est petit comparé à la sensibilité des capteurs, mais coûteux en temps de calcul). Nous utilisons un schéma d'Euler de premier ordre pour l'intégration numérique.

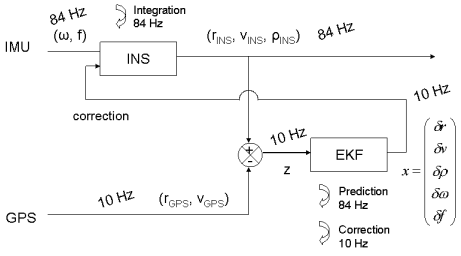


Figure 4: *Couple GPS-INS*

Dans le but de corriger les dérives de la centrale inertielle,

nous utilisons un filtre de Kalman étendu (EKF) avec corrections par feedback (Grewal et al. 2001), comme représenté sur la figure 4.

3.2. Géo-référencement des données

Le laser fournit des données sous forme de distance et d'angle dans un plan mobile avec le véhicule. Il est alors possible de combiner ces données avec l'information de localisation de la voiture pour produire un nuage de points 3D dans un référentiel commun terrestre ([ANL*03]) (voir la figure 5).

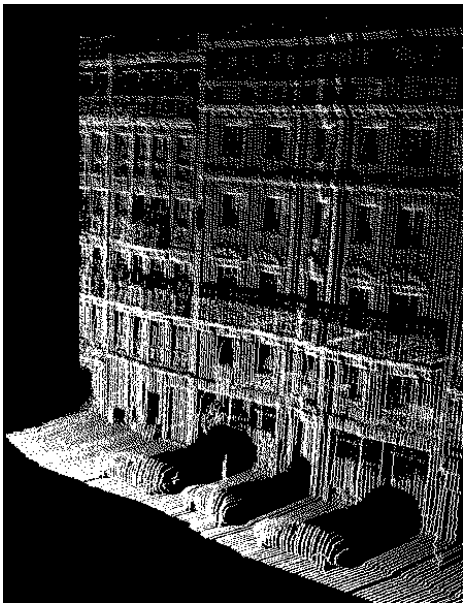


Figure 5: *Nuage de points 3D d'une rue*

3.3. Précision des points

Nous devons faire la distinction entre la précision relative et absolue des données. La précision absolue est liée aux biais contenus dans les mesures alors que la précision relative est seulement liée au bruit aléatoire affectant les données.

Dans notre cas, nous avons plusieurs sources de précision absolues et relatives. Pour la fonction de localisation, la meilleure précision relative est donnée par la centrale inertielle alors qu'elle a une très mauvaise précision absolue à cause des dérives corrigées par le GPS. Au contraire, la meilleure précision absolue est donnée par le GPS (qui pourrait être plus précise avec un DGPS). A partir d'une série de tests de modélisation d'une route, nous avons mesuré une précision relative de nos données avec une déviation standard de $\sigma \approx 1mm$ et une précision absolue (biais) inférieure à 4 m.

Table 2: *Précision estimée des points 3D*

Précision	Relative (déviation σ)	Absolue (biais)
Points 3D	$\approx 5cm$ (direction du laser)	$< 4m$ (GPS disponible)

3.4. Triangulation

La triangulation s'effectue entre deux profils adjacents à l'aide d'un algorithme 2D itératif de Delaunay. Pour cela, il est nécessaire de paramétriser les points 3D en points 2D avec les coordonnées cylindriques en r et θ (cf. figure 6). Le résultat sur une scène urbaine est visible sur la figure 7.

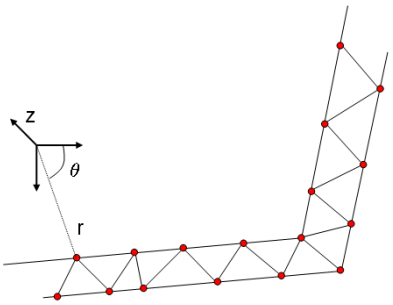


Figure 6: *Schéma de la triangulation*

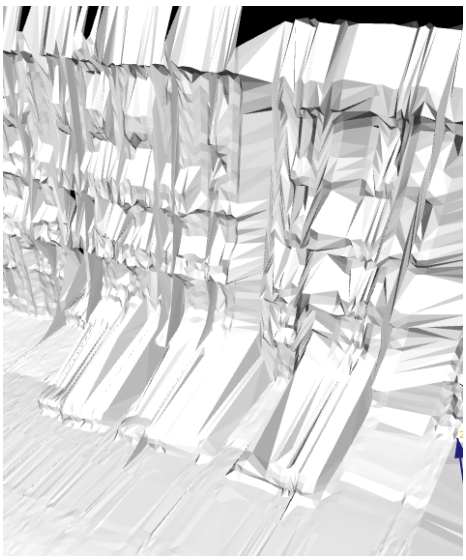


Figure 7: *Triangulation d'une rue*

4. L'ajout de la couleur et des textures

4.1. Caméra et Calibrage

Dans cette partie, nous voulons déterminer la transformation rigide entre le repère de la caméra et celui du laser, c'est à dire une matrice de rotation Φ et un vecteur de translation Δ. Si P_c est un point dans le système de coordonnées de la caméra et P_l le même point dans le système de coordonnées du laser, nous avons :

$$P_l = \Phi * P_c + \Delta \quad (1)$$

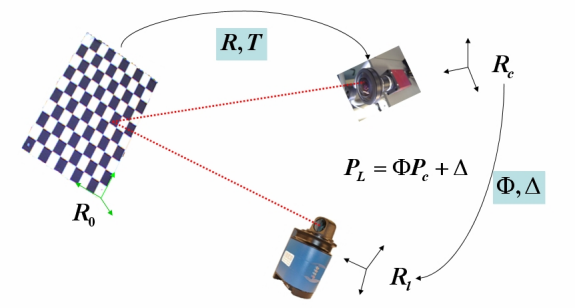


Figure 8: *Calibrage Caméra - Laser*

Nous utilisons la méthode décrite par Pless et Zhang [PZ04] et améliorée par [DKF05]. L'idée de base est d'utiliser une mire de calibrage plane sous forme d'échiquier comme on le voit sur la figure 8. Les points du laser doivent tomber sur la mire plane dont la position est estimée par les images de la caméra. Nous avons alors une contrainte géométrique sur la transformation rigide entre le système de coordonnées de la caméra et le système de coordonnées du laser. Nous effectuons les mesures de chaque capteur au moins une quinzaine de fois et à chaque fois, la position de la mire est différente. Pour ce processus de calibrage, nous utilisons une lentille "classique" parce que l'on a juste besoin des paramètres extrinsèques de la caméra. Tout d'abord, nous effectuons le Calibrage de Tsai [Tsa87] pour obtenir la position de la mire dans le repère de la caméra. Ces positions sont définies comme des plans : pour la $i$ème pose, N_i est le vecteur normal au plan et d_i la distance à l'origine. Nous utilisons l'algorithme optimisé de Levenberg Marquardt pour minimiser les erreurs de reprojection des points laser pour toutes les configurations. La fonction à minimiser est :

$$\sum_i \sum_j \left(\frac{N_i}{\|N_i\|} (\Phi^{-1} P_{ij} - \Delta) - d_i \right)^2 \quad (2)$$

Pless et Zhang [PZ04] ont proposé une solution algébrique qui peut être utilisé en tant que solution initiale et la convergence est alors très rapide (approximativement 10-20 itérations). Il est important de noter que lors du processus, la lentille utilisée n'est pas une lentille grand angle. En théorie, cela n'a pas de conséquence puisque nous utilisons seulement les paramètres extrinsèques.

4.2. Ajouter une lentille fish-eye

Figure 9: *Exemple d'une image fish-eye*

Nous avons équipé la caméra d'un objectif fish-eye dans le but d'avoir un grand angle d'ouverture. En conséquence, les conditions de Gauss ne sont plus valables : le rayon de lumière traverse la lentille loin de son centre et la lumière n'est plus quasi-parallèle à l'axe optique. C'est pourquoi le modèle pin-hole ne peut plus être utilisé, nous avons besoin d'un autre modèle de projection.. Nous avons décidé de prendre le modèle équidistant où la position du point projeté dépend de θ et non plus de $tan(\theta)$ comme dans le modèle pin-hole (cf. figure 10).

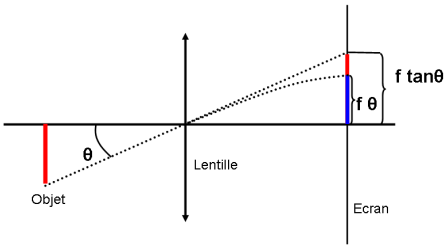


Figure 10: *Différents modèles de projection*

Après cette projection, nous devons passer en unité pixellique. Nous avons besoin de quatre paramètres, u_0 et v_0 la position du centre et la taille μ_x et μ_y de chaque pixel. Si nous considérons un point 3D $P(\rho, \theta, \varphi)$ dans le repère caméra R_c, on obtient la position du point projeté en pixels dans R_p par :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{pmatrix} \mu_x & 0 \\ 0 & \mu_y \end{pmatrix} r(\theta) \begin{pmatrix} cos(\varphi) \\ sin(\varphi) \end{pmatrix} \quad (3)$$

Notre méthode de calibrage est basée sur la forme d'une image fish-eye (cf. figure 9). Nous supposons que la projection du monde sur le capteur CCD est un cercle. Les autres pixels sont noirs ce qui permet de déterminer facilement quel pixel appartient au cercle. Nous utilisons un filtre de Canny pour récupérer la bordure du cercle et différentes méthodes classiques de traitement de l'image pour estimer les paramètres du cercle comme la transformation circulaire de Hough [HAM01].

Le centre du cercle est la position de la projection du centre optique de la caméra sur l'image (u_0, v_0) et nous utilisons le rayon du cercle r_c pour estimer la focale f. La distance r entre le centre de l'image et la position d'un pixel est proportionnelle à l'angle d'incidence θ, $r = f\theta$ (cf. figure 10). Nous pouvons effectuer le même calcul pour la distance maximale, c'est à dire le rayon du cercle :

$$f = \frac{r_c}{\theta_{max}}. \quad (4)$$

θ_{max} est égal à la moitié de l'angle d'ouverture de l'objectif fish-eye, donné par le constructeur. La taille des cellules CCD (μ_x et μ_y) sont donnés par le calibrage précédente.

4.3. Points colorés

Nous avons tous les paramètres nécessaires pour projeter un point du repère laser sur une image de la caméra fish-eye et ainsi récupérer la couleur du point 3D correspondant. Sachant que les traitements sont faits en temps réel, à chaque donnée laser (c'est à dire un profil complet de 1080 points), nous choisissons l'image qui a été acquise à une date la plus proche de la date d'acquisition des données laser. Nous projetons ensuite chaque point du repère caméra dans l'image (à l'aide du calibrage repère caméra-laser) et l'on affecte la couleur du pixel au point 3D. Grâce à la géo-localisation, nous pouvons convertir les coordonnées des points laser de chaque profil en points 3D dans un repère monde fixe. Nous obtenons alors un nuage de points 3D colorés (cf. figure 11).

4.4. Textures

4.4.1. La création de cartes de textures

Avec un modèle 3D triangulé pour représenter l'environnement urbain, nous pouvons y ajouter des textures en utilisant la caméra embarquée. La vitesse de la voiture est typiquement de l'ordre de 5km/h et le télémètre laser donne 1080 points à chaque profil à une fréquence de 10Hz. Deux profils adjacents permettent de créer environ

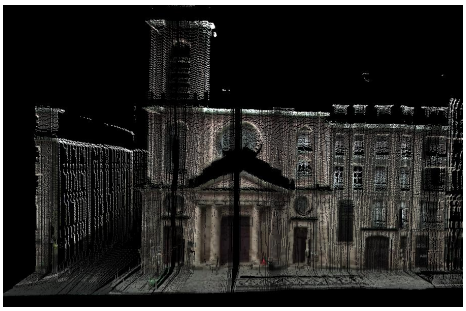


Figure 11: *Résultats de la colorisation des points laser à l'aide de la caméra fish-eye*

300 triangles avec un algorithme de décimation et une triangulation de Delaunay. La voiture aura alors avancé de 15cm durant ce temps. Nous travaillons avec les points définissant les triangles dans le système fixe de coordonnées monde. La position de la voiture est représentée par des matrices de rotation et de translation qui transforment les coordonnées monde en coordonnées caméra. En pratique, la fusion GPS/INS nous donne la transformation entre les coordonnées laser et les coordonnées monde. Nous utilisons donc la transformation rigide entre les coordonnées laser et caméra calculée lors de l'étape de calibrage.

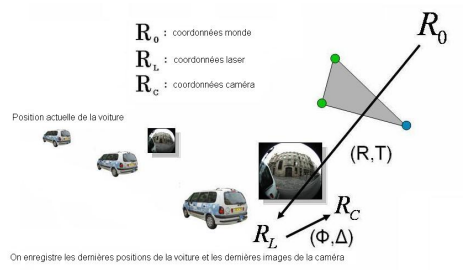


Figure 12: *Projection des textures sur les triangles*

Le même algorithme itératif est appliqué pour chaque triangle. Avec les coordonnées monde des points des triangles, nous calculons les coordonnées de ces mêmes points dans un repère caméra choisi. Dans notre application, sachant que les points d'un triangle appartiennent à des profils différents mais adjacents, nous avons choisi de garder l'image dont la date d'acquisition est la plus proche de la date d'acquisition des profils. Cela nous donne une image avec la meilleure résolution possible. Tout ces calculs sont effectués de façon retardé (les calculs se font à la même vitesse que l'acquisition des données mais on est obligé de garder un délai de quelques profils pour le calcul de la triangulation). Cela requiert de conserver les dernières positions de la voiture ainsi que les dernières images de la caméra. Après cela, nous projetons les 3 points extrémités de chaque triangle sur l'image fish-eye à l'aide du modèle de projection

équidistant. Nous enregistrons les textures dans des cartes de textures et OpenGL peut alors afficher les triangles et leurs textures associées en temps réel.

4.4.2. Optimisations

Nous enregistrons les textures dans des cartes de textures (fichiers BMP) en entourant les triangles par des rectangles. Pour gagner de la mémoire (ceci est important pour afficher les modèles plus rapidement), nous effectuons une classification des rectangles en les regroupant suivant leur taille [RCM99]. Avec la classification, nous avons besoin de seulement 108 cartes de textures à la place des 302 fichiers à l'origine pour le modèle d'une rue (figure 13).

(a) (b)

Figure 13: *Carte de textures avec (a) et sans (b) classification*

Nous utilisons une technique pour ajouter de la précision dans l'estimation des coordonnées de texture. Nous travaillons avec des flottants à la place d'entiers pour le calcul de la position en pixels des points projetés dans chaque image. Par exemple dans la figure 14, nous gardons le triangle bleu à la place du triangle noir pour les textures. Avec une image fish-eye 776x580, 10cm sur la façade d'un immeuble à une distance de 5m sera représentée par 3 pixels sur l'image. On peut ainsi comprendre l'importance de passer à une résolution sub-pixellique dans le calcul des coordonnées des textures (inspiré de [NSO00]).

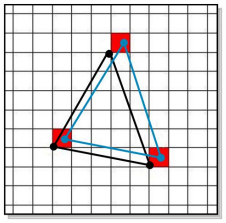


Figure 14: *Résolution sub-pixellique pour les textures*

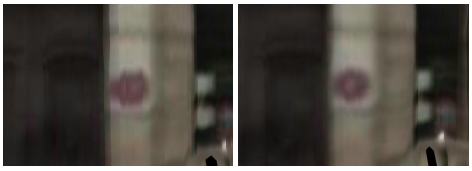
4.4.3. Améliorations de la Visualisation

Pour améliorer les résultats de visualisation, nous avons ajouté un filtrage linéaire des textures à la frontière des triangles. En effet, nous pouvons voir apparaître des différences de luminosité entre des textures de deux triangles adjacents. Des erreurs sur les paramètres de la caméra peuvent aussi donner des frontières trop visibles entre les triangles texturés. En appliquant un filtre linéaire sur les bords des textures des triangles adjacents, nous obtenons une frontière qui sera moins visible [NB95] (voir la figure 16).

(a) Image provenant de la caméra

(b) Modèle 3D texturé

Figure 15: *Comparaison entre une image de la caméra et le modèle 3D texturé*



(a) Détail sans filtres (b) Détail avec filtres

Figure 16: *Détails du modèle 3D texturé*

4.5. Analyse des résultats

Le modèle 3D que nous venons de texturer a encore certaines limites : tout d'abord, il y a quelques erreurs produites lors de l'enregistrement (on voit des discontinuités dans le modèle : voir la figure 11). Cela est du à une erreur sur la localisation qui s'avère cruciale dans la construction du modèle texturé : en effet, le positionnement de la voiture est connu avec une précision relative de 5cm (cela est du au fait que

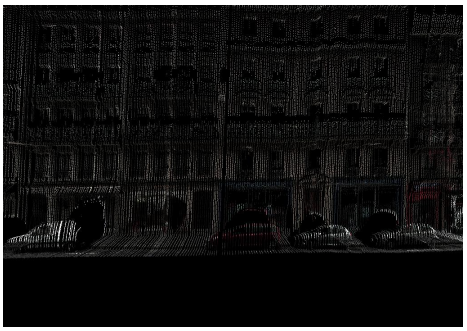


Figure 17: *Nuage de Points 3D coloré*

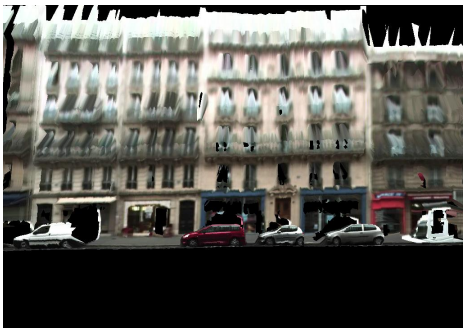


Figure 18: *Modèle 3D texturé*

le GPS n'est pas très fiable surtout en ville) et que la centrale inertielle ne permet pas de connaître la position de la voiture sur une longue distance. Une solution serait d'utiliser un GPS différentiel qui permettrait d'avoir un facteur 10 sur la précision absolue de la voiture. Nous remarquons aussi des bavures des textures sur les hauts des immeubles (voir la figure 18). Ceci est principalement causé par la faible résolution de la caméra qui est de 400 000 pixels (10cm sur la façade d'un immeuble à 5m de distance représentent 3 pixels sur l'image). C'est en augmentant cette résolution (projet en cours) que l'on pourra mesurer les autres facteurs à améliorer dans le processus de texturation. Nous travaillons sur plusieurs autres améliorations possibles de nos modèles par le développement de nouvelles techniques de rendu de la scène (en texturant les objets non visibles par le laser, en enlevant les objets de premier plan).

5. Conclusion

Nous avons présenté un nouveau système temps réel, embarqué sur un véhicule, de construction de modèles texturés réalistes de l'environnement. Ce système est basé sur le couplage et le calibrage de deux capteurs, un télémètre laser et une caméra équipée d'un objectif fish-eye. Les résultats montrent que la texturation temps réel est possible et peut être améliorée.

References

[ANL*03] ABUHADROUS I., NASHASHIBI F., LAURGEAU C., CHINCHOLE M., GOULETTE F.: Multisensor data fusion for land vehicle localization using rtmaps. In *Intelligent Vehicles Symposium* (2003).

[ASG*01] ALLEN P., STAMOS I., GUEORGUIEV A., GOLD E., BLAER P.: Avenue: Automated site modeling in urban environments. In *In Proc. of 3rd Conference on Digital Imaging and Modeling (3DIM'01)* (2001).

[DKF05] DUPONT R., KERIVEN R., FUCHS P.: An improved calibration technique for coupled single row telemeter and ccd camera. In *3D Digital Imaging and Modeling (3DIM)* (2005).

[FB98] FARRELL J., BARTH M.: The global positioning system and inertial navigation.

[FZ03] FRUEH C., ZAKHOR A.: 3d model generation for cities using aerial photographs and ground level laser scans. *Computer Vision and Pattern Recognition* (June 2003).

[HAM01] HADAD R. M., ARAUJO A. D. A., MARTINS P.: Using the hough transform to detect circular forms in satelliteimagery. In *Brazilian Symposium on Computer Graphics and Image Processing* (2001).

[NB95] NIEM W., BROSZIO H.: Mapping texture from multiple camera views onto 3d-object models for computer animation. In *International Workshop on Stereoscopic and Three Dimensional Imaging* (1995).

[NSO00] NAKAMURA K., SAITO H., OZAWA S.: Generation of 3d model with super resolved texture from image sequence. In *Systems, Man, and Cybernetics* (2000).

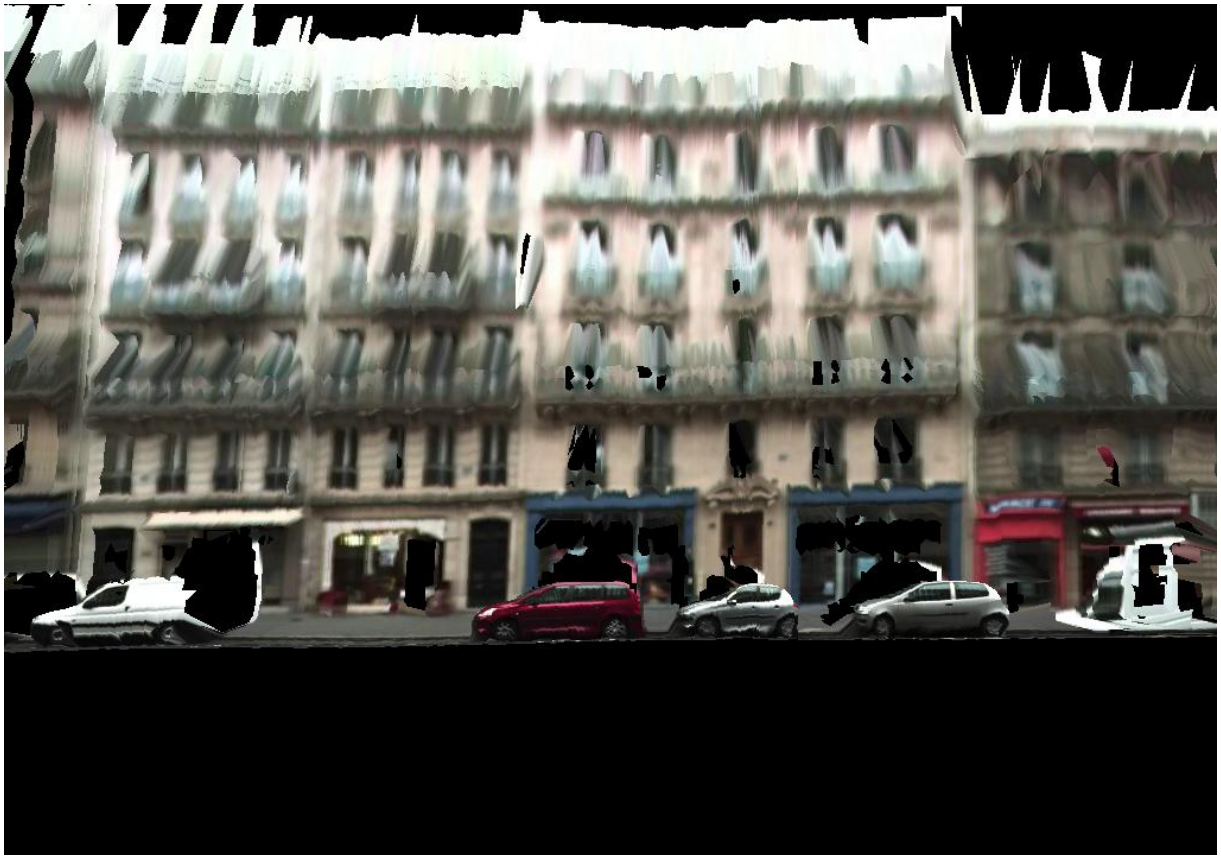
[PZ04] PLESS R., ZHANG Q.: Extrinsic calibration of a camera and laser range finder. In *Intelligent Robots and Systems(IROS)* (2004).

[RCM99] ROCCHINI C., CIGNONI P., MONTANI C.: Multiple textures stitching and blending on 3d objects. In *Proc. Eurographics Workshop Rendering* (1999).

[SA00] STAMOS I., ALLEN P.: 3d model construction using range and image data. In *In. Proc Int. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2000).

[Tsa87] TSAI R. Y.: A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation RA-3,* (August 1987), 323–344.

[ZS03] ZHAO H., SHIBASAKI R.: A vehicle-borne urban 3d acquisition system using single-row laser range scanners. *IEEE Trans. SMC Part B: Cybernetics 33* (August 2003).



Calcul des angles charnières à partir d'images numériques tournées

Yohan Thibault [1] Yukiko Kenmochi [1,2]

[1]Université Paris Est - Marne la Vallée
[2]Université Paris Est - Marne la Vallée

Abstract

Nouvel et Rémila ont proposés une notion de rotation discrète basée sur les angles charnières pour les images numériques en 2D [NR06]. Les angles charnières sont connus comme étant des angles particuliers déterminés par une image numérique donnée, tel que quelque soit l'angle compris entre deux angles charnières consécutifs le résultat renvoyé par RED soit le même. Ils ont montrés que tout angle charnière peut être représenté de manière unique sous la forme d'un triplet d'entiers, et donc qu'une rotation d'image numérique pouvait être réalisée avec un calcul uniquement en entiers. Dans cet article, nous montrerons, dans un premier temps, comment obtenir l'angle charnière directement inférieur à un angle Euclidien donné. Ensuite nous présenterons un algorithme efficace pour effectuer une rotation discrète à partir de l'angle charnière précédemment trouvé, avec le même résultat que l'angle Euclidien. Nous montrerons enfin comment obtenir, à partir d'une paire d'images numériques, le triplet d'entiers qui définit la relation géométrique entre ces deux images.

1. Introduction

La rotation est l'une des transformations de base pour les images numériques, elle est nécessaire dans de nombreuses applications pour l'imagerie numérique. Le but de cet article est de trouver la relation géométrique existante entre deux images numériques, dont on sait qu'elle représente le même objet, c'est à dire la transformation pour passer de l'une à l'autre. Dans le plan Euclidien $\mathbb{R}^2$, la rotation est définie comme suit. La rotation Euclidienne d'angle γ deplace un point (x_1, y_1) de $\mathbb{R}^2$ en un point (x_2, y_2) de $\mathbb{R}^2$ tel que

$$x_2 = x_1 \cos\gamma - y_1 \sin\gamma, \quad (1)$$

$$y_2 = x_1 \sin\gamma + y_1 \cos\gamma. \quad (2)$$

Considérons maintenant la rotation dans le plan discret $\mathbb{Z}^2$. Du fait que les valeurs x_2 et y_2 dans (1) et (2) ont peu de chance d'être entières même si x_1 et y_1 sont des entiers, nous ne pouvons appliquer (1) et (2) au points de $\mathbb{Z}^2$ directement. L'approximation la plus simple d'une telle rotation Euclidienne est l'utilisation de la fonction arrondie, telle que

$$P_2 = \lfloor P_1 \cos\gamma - Q_1 \sin\gamma + \frac{1}{2} \rfloor, \quad (3)$$

$$Q_2 = \lfloor P_1 \sin\gamma + Q_1 \cos\gamma + \frac{1}{2} \rfloor, \quad (4)$$

où un point (P_1, Q_1) dans $\mathbb{Z}^2$ est déplacé en (P_2, Q_2) dans $\mathbb{Z}^2$. Une telle rotation entre deux points discrets de $\mathbb{Z}^2$ est appelé rotation Euclidienne discrétisée d'angle γ, abrégé par la suite RED. Cependant, RED utilise les fonctions sinus et cosinus qui elles-mêmes utilisent des nombres réels ce qui constitue un problème. De plus en pratique, les erreurs d'approximation de ces fonctions peuvent causer des erreurs dans les résultats de RED.

Dans cet article, nous avons pour but d'éviter de tels erreurs. A cette fin, nous avons développé un algorithme de rotation garantissant d'obtenir les mêmes résultats que RED. De plus notre algorithme nous permet d'effectuer la rotation d'images numériques en utilisant uniquement des calculs avec des entiers, i.e. sans utiliser de sinus ou de cosinus. Une rotation n'utilisant que des calculs avec des entiers sera appelée dans cet article une rotation discrète.

Il existe déjà des travaux sur des rotations discrètes. Andrès a étudié quelques rotations discrètes se basant sur les rotations par cercles, par droites, et par droites Pythagoriciennes, respectivement [And92]. Toutes ces rotations utilisent seulement des entiers durant leurs calculs, mais leur résultats sont rarement les mêmes que que ceux obtenus par RED. Dans [And96], il a proposé une autre rotation dis-

crète appelée rotation quasi transvection. Bien que les résultats obtenus soient meilleurs que ceux des précédentes méthodes, cette rotation discrète ne garantit toujours pas d'obtenir les mêmes résultats que RED. Dans [NR06], Nouvel et Rémila ont, de leur côté, proposé une nouvelle rotation discrète se basant sur les angles charnières. Les angles charnières sont connus comme étant des angles particuliers déterminés par une image numérique donnée, tel que quelque soit l'angle compris entre deux angles charnières consécutifs le résultat renvoyé par RED soit le même. En d'autres termes, les angles charnières correspondent au discontinuités de (3) et (4). Nouvel et Rémila ont montré que tout angle charnière peut être représenté de manière unique sous la forme d'un triplet d'entiers, et donc qu'une rotation d'image numérique pouvait être réalisé avec un calcul uniquement en entiers. De plus leur algorithme donne les mêmes résultats que RED. En conséquence, les angles charnières, désignés par le triplets d'entiers leur correspondant, donnent suffisamment d'informations pour effectuer toutes les rotations d'images possibles.

Dans cet article, nous aborderons la rotation discrète basée sur les angles charnières. Nous introduirons dans un premier temps la notion d'angle charnière ainsi que certaines de leurs propriétés. Ensuite, nous montrerons comment obtenir un angle charnière à partir d'un angle donné, puis comment à partir du triplet d'entiers calculer de manière efficace la rotation d'une image numérique. Parce que l'intérêt dans [NR06] était plus d'étudier la dynamique de la rotation discrète que les angles charnières en eux mêmes, leur algorithme pour la rotation d'images numériques nécessite de calculer tout les angles charnières possibles pour une image donnée; la complexité en temps est alors de $O(n^3 \log(n))$ où n est la taille du plus grand coté de l'image. Nous améliorons leur algorithme pour notre objectif en en développant un avec une complexité de $O(n^2 \log(n))$. Nous présenterons enfin comment obtenir, à partir d'une paire d'images numériques, représentant le même objet, le triplet d'entiers qui définit la relation géométrique entre ces deux images.

2. Les angles charnières

Considérons les points de $\mathbb{Z}^2$ comme les centres des pixels, leurs rotations se feront autour d'un centre à coordonnées entières. Les angles charnières sont des angles particuliers qui déplacent par rotation des points de $\mathbb{Z}^2$ sur la frontière entre deux pixels. Dans cette section, nous donnons la définition des angles charnières ainsi que leurs propriétés. Nous introduisons aussi les angles Pythagoriciens qui sont fortement liés aux angles charnières.

2.1. Définition des angles charnières

Soit $\vec{x}$ un point dans $\mathbb{R}^2$ tel que $\vec{x} = (x, y)$. On dit que $\vec{x}$ possède une coordonnée demie entière si $x - \frac{1}{2} \in \mathbb{Z}$ ou/et $y - \frac{1}{2} \in \mathbb{Z}$. L'ensemble des points qui possèdent au moins une coordonnée demie entière se note $\mathscr{H}$ et s'appelle la demie grille. Plus simplement, $\mathscr{H}$ représente l'ensemble des points se trouvant sur la frontière des pixels dont le centre sont des points de $\mathbb{Z}^2$.

Definition 1 Un angle α est un angle charnière si il existe au moins un point de la grille de $\mathbb{Z}^2$ tel que son image par la rotation Euclidienne d'angle α appartienne à $\mathscr{H}$.

De même que $\mathscr{H}$ peut être vu comme la discontinuité de la fonction arrondie dans (3) et (4), les angles charnières peuvent être vu comme la discontinuité de RED. Plus simplement, les angles charnières déterminent la transition durant la rotation d'un point de la grille $\mathbb{Z}^2$ d'un pixel à un pixel adjacent.

La proposition suivante est importante car elle montre que tout angles charnières peut être exprimé à partir d'un triplet d'entiers.

Proposition 1 Un angle α est un angle charnière si il existe un triplet d'entiers (P, Q, K) tel que

$$2Q\cos\alpha + 2P\sin\alpha \;=\; 2K + 1. \tag{5}$$

avec $(K + \frac{1}{2})^2 < P^2 + Q^2$

La preuve est donné dans [NR06]. Géométriquement, un angle charnière α est composé de deux segments partants de l'origine et se terminant l'un sur un point de la grille entière (P, Q) et l'autre sur un point de la demi grille $(K + \frac{1}{2}, \lambda)$ comme montré sur la Figure 1(gauche). On peut déduire de cette proposition que tout les calculs liés aux angles charnières peuvent être effectué uniquement avec des entiers. Dans la suite, α indiquera toujours un angle charnière, et nous noterons $\alpha(P, Q, K)$ l'angle charnière généré par le triplet d'entiers (P, Q, K).

Posons $\lambda = \sqrt{P^2 + Q^2 - (K + \frac{1}{2})^2}$, les équations suivantes peuvent facilement être déduite de (5) et de la Figure 1(gauche).

$$\cos\alpha = \frac{P\lambda + Q(K + \frac{1}{2})}{P^2 + Q^2}, \qquad \sin\alpha = \frac{P(K + \frac{1}{2}) - Q\lambda}{P^2 + Q^2}. \tag{6}$$

Notons que l'on peut avoir un point de la demi grille $(\lambda, K + \frac{1}{2})$, à la place de $(K + \frac{1}{2}, \lambda)$. Dans ce cas la, les équations précédentes deviennent $\cos\alpha = \frac{Q\lambda + P(K + \frac{1}{2})}{P^2 + Q^2}$, et $\sin\alpha = \frac{Q(K + \frac{1}{2}) - P\lambda}{P^2 + Q^2}$. Les symétries sur les angles charnières sont importantes car elles permettent de réduire les rotations au premier quadrant du cercle, tel que $\alpha \in [0, \frac{\pi}{2}]$.

Corollaire 1 Chaque triplet d'entiers (P, Q, K) correspond à quatre angles charnières symétriques $\alpha + \frac{\pi k}{2}$ avec $k = 0, 1, 2, 3$.

Dans le but de distinguer le cas $(K + \frac{1}{2}, \lambda)$ du cas $(\lambda, K + \frac{1}{2})$, nous changeons le signe de K; nous utilisons $\alpha(P, Q, K)$ pour le cas $(K + \frac{1}{2}, \lambda)$, et $\alpha(P, Q, -K)$ pour le cas $(\lambda, K +$

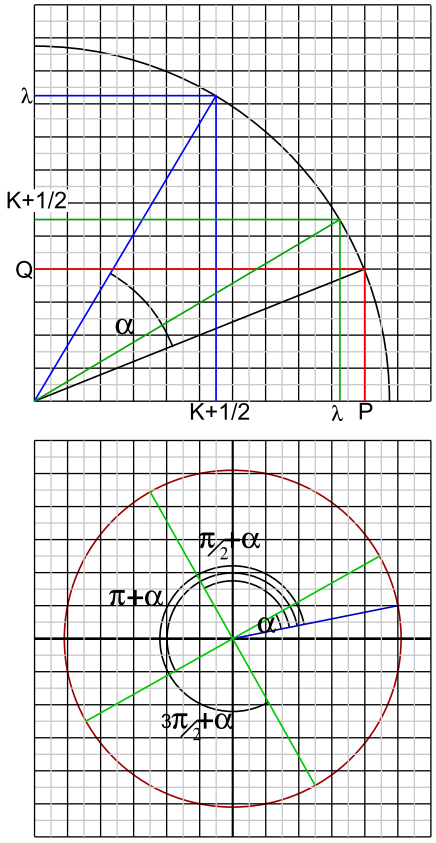


Figure 1: *Un angle charnière* $\alpha(P,Q,K)$ *(gauche) et quatre angles charnières symétriques (droite).*

$\frac{1}{2}$). Notons que les symétries nous permettent de restreindre α à l'intervalle $[0,\frac{\pi}{2}]$, nous savons donc que K est toujours positif.

2.2. Propriétés relative aux angles Pythagoricien

Du fait que les angles charnières sont fortement liés aux angles Pythagoriciens, les propriétés des angles Pythagoriciens sont nécessaires pour démontrer certaine propriétés des angles charnières. Par conséquent, nous donnons maintenant la définitions des angles Pythagoriciens et leurs propriétés.

Definition 2 Un angle θ est appelé Pythagoricien si $\cos\theta$ et $\sin\theta$ appartiennent à l'ensemble des nombres rationnels $\mathbb{Q}$.

Nous pouvons déduire de la Définition 2 que tout angle Pythagoricien θ est défini par un triplet d'entier (a,b,c) tel que

$$\cos\theta = \frac{a}{c}, \qquad \sin\theta = \frac{b}{c}. \tag{7}$$

Dans la suite, θ indiquera un angle Pythagoricien. Le lemme suivant est nécessaire pour la proposition qui le suit.

Lemme 1 Soit (a,b,c) un triplet d'entiers générant un angle Pythagoricien avec $\mid a \mid < \mid b \mid < \mid c \mid$. Si $\gcd(a,b,c) = 1$, alors c est impair.

Proof Nous supposons que c est pair tel que $c = 2d$ avec d dans $\mathbb{Z}$. Nous obtenons alors a et b tout deux impair car $a^2+b^2 = c^2 = (2d)^2$. Sinon nous aurions $\gcd(a,b,c) = 2n$ avec $n \in \mathbb{Z}$, ce qui est contradictoire. Nous pouvons donc Poser $a = 2e+1$ et $b = 2f+1$ avec $e,f \in \mathbb{Z}$, en remplacant nous obtenons $(2e+1)^2+(2f+1)^2 = 4d^2$, ce qui peut être récrit $e^2+e+f^2+f+\frac{1}{2} = d^4$. Cela implique que d ne peut pas appartenir à $\mathbb{Z}$. Nous pouvons donc conclure que c est impair. □

Si $\gcd(a,b,c) = i$, alors $\gcd(\frac{a}{i},\frac{b}{i},\frac{c}{i}) = 1$ et le triplet d'entiers $(\frac{a}{i},\frac{b}{i},\frac{c}{i})$ génère le même angle Pythagoricien que (a,b,c).

Proposition 2 Soit E_h l'ensemble des angles charnières et E_p l'ensemble des angles Pythagoriciens. Nous avons alors $E_h \bigcap E_p = \emptyset$.

Proof Supposons qu'il existe un angle α tel que $\alpha \in E_h$ et $\alpha \in E_p$. Du fait que α appartient à E_p, il existe alors un triplet d'entiers (a,b,c) générateur de l'angle α avec $\gcd(a,b,c) = 1$. Par substitution de (7) dans (6), on obtient

$$2\frac{Qa+Pb}{c} = 2K+1 \tag{8}$$

d'où on dérive $2\frac{Qa+Pb}{c} \in \mathbb{Z}$. On sait grâce au lemme 1 que c est impair, on déduit donc que $\frac{Qa+Pb}{c} \in \mathbb{Z}$. Cependant, cela contredit le fait que pour toute pair n,m appartenant à $\mathbb{Z}$, on ne peut pas avoir $2n = 2m+1$. On peut en déduire que α ne peut pas appartenir à E_h et E_p simultanément. □

Cette proposition montre qu'un angle charnière ne peut pas tourner un point de la grille sur un point(x,y) tel que $x = i+\frac{1}{2}, y = j+\frac{1}{2}$, avec $(i,j) \in \mathbb{Z}^2$.

3. Calcul d'angles charnières à partir d'un angle Pythagoricien

Dans cette section, nous proposons une méthode pour calculer l'angle charnières correspondant à un angle donné afin d'effectuer la rotation d'une image numérique donnée. Nouvel et Rémila ont présenté une méthode pour calculer tout les angles charnières pour un point de la grille ou un pixel dans une image numérique [NR06]. Leur méthode peut être adaptée pour trouver l'angle charnière qui nous intéresse à savoir la borne inférieure d'un angle donné. La complexité en temps serait alors de $O(n\log(n))$ avec n le nombre d'angle charnière existant pour le point de la grille donné. Notons que n dépend des coordonnées du point de la grille. Dans la sous-section 3.1, nous améliorons leur méthode en utilisant une structure d'arbre sur les angles charnières, la complexité de notre méthode est alors de $O(\log(n))$. Dans la sous-section 3.2, nous présentons une méthode pour trouver l'angle charnière constituant la borne inférieure d'un angle pour une image numérique donnée, c'est à dire pour

l'ensemble des pixels de l'image. Finalement, nous donnons un algorithme de rotation discrète à partir de l'angle charnière précédemment calculé. Cet algorithme donne le même résultat que RED avec l'angle de rotation original. Notons que, dans cet article, l'angle donné en entré dans l'algorithme est un angle Pythagoricien, comme dans [NR06]. Cependant, il est possible d'utiliser un angle Euclidien car il existe une méthode en temps linéaire $O(m)$ pour approximer un angle Euclidien par un angle Pythagoricien avec une précision de $\frac{1}{10^m}$ [Ang88].

3.1. Calcul de l'angle charnière constituant la borne inférieure pour un point

Pour chaque point de la grille $\vec{p} = (P,Q)$ dans $\mathbb{Z}^2$, il y a au plus $n = \lfloor \sqrt{P^2+Q^2} + \frac{1}{2} \rfloor$ angles charnières différents [NR06]. Cela implique qu'il existe une séquence de $K_i, i = 0,1,..,n-1$ dans $\mathbb{Z}$, avec $0 \leq K_i < n$ pour tout $\vec{p}$. Il est possible de comparer les angles charnières deux à deux, nous pouvons donc obtenir un ensemble croissant totalement ordonné $\{\alpha_1(P,Q,K_1), \alpha_2(P,Q,K_2), ..., \alpha_{max}(P,Q,K_{max})\}$ tel que $\alpha_1 < \alpha_2 < ... < \alpha_{max}$. Soit θ un angle Pythagoricien donné, pour trouver l'angle charnière constituant la borne inférieur α_i tel que $\alpha_i < \theta < \alpha_{i+1}$, nous utilisions une structure d'arbre. Par une recherche binaire, nous pouvons trouver α_i en un temps $O(\log(n))$, en supposant que l'on puisse faire la comparaison entre un angle charnière et un angle Pythagoricien en temps constant. Cet algorithme est décrit dans la Figure 1.

Data: Point $\vec{p}(P,Q)$, angle Pythagoricien θ
Result: $\alpha(P,Q,K)$
var $K_{max} = \lfloor \sqrt{P^2+Q^2} - 1 \rfloor$;
var $K_{min} = 0$;
var $K_{min} = 0$;
while $K_{max} + K_{min} \neq 1$ **do**
  if $\alpha(P,Q,K) > \theta$ **then**
    $K_{max} = K$;
  else
    $K_{min} = K$;
  end
  $K = \lfloor \frac{K_{max}+K_{min}}{2} \rfloor$;
end
return $\alpha(P,Q,K)$
Function `Calcul de l'angle charnière`

La proposition suivante montre que l'on peut faire la comparaison entre un angle charnière et un angle Pythagoricien en temps constant.

Proposition 3 Soit α un angle charnière et θ un angle Pythagoricien. On peut vérifier si $\alpha > \theta$ en temps constant et avec un calcul en entier.

Proof Soit $\alpha(P,Q,K)$ un angle charnière dans $[0, \frac{\pi}{2}]$ et $\theta(a,b,c)$ un angle Pythagoricien dans $[0, \frac{\pi}{2}]$. A partir de (6) et (7), on obtient

$$\cos\alpha - \cos\theta = \frac{P(K+\frac{1}{2}) + Q\lambda}{P^2+Q^2} - \frac{a}{c}.$$

Si θ est plus grand que α, $\cos\alpha - \cos\theta > 0$, alors

$$cP(4K+2) - 4a(P^2+Q^2) > -4cQ\lambda. \quad (9)$$

Du fait que c, Q, λ sont positifs, la partie droite de (9) est toujours négative. Donc si la partie gauche de (9) est positive ou nulle, alors $\theta > \alpha$. Sinon, on élève (9) au carré, et il ne reste plus qu'a vérifier l'inégalité suivante :

$$\left[cP(4K+2) - 4a(P^2+Q^2)\right]^2 < 16c^2Q^2\lambda^2. \quad (10)$$

Par définition $\lambda = \sqrt{P^2+Q^2-(K+\frac{1}{2})^2}$, donc $16\lambda^2$ est un entier et la partie droite de l'équation (10) n'est composé que de valeurs entières. Nous pouvons donc vérifier (10) avec un calcul uniquement en entier. Si l'inéquation est vraie alors $\theta > \alpha$; sinon $\alpha > \theta$. □

Notons la particularité des rotations d'angles $\frac{\pi}{2}$ et de ses multiples. Si l'angle de rotation est égal à $\frac{\pi}{2}, \pi, \frac{3\pi}{2}$, il suffit d'inverser et/ou de changer le signe des coordonnées x et/ou y. Pour cette raison nous pouvons restreindre l'angle de rotation en entré de l'algorithme θ à $0 < \theta < \frac{\pi}{2}$.

3.2. Calcul de l'angle charnière constituant la borne inférieure pour une image

Dans cette sous-section, nous présentons un algorithme, basé sur le précédent, pour calculer l'angle charnière constituant la borne inférieure pour un angle Pythagoricien θ donné et pour une image numérique donnée S se résumant en m points de la grille tel que $S = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_m\}$. La sortie est un triplet d'entiers qui représentent l'angle charnière. L'algorithme calcule l'angle charnière correspondant à chacun des m points de la grille, puis les tries pour conserver le plus grand. Plus précisément, il calcul dans un premier temps l'angle charnière inférieur à l'angle θ pour le premier point de S, et il le stocke comme référence. Ensuite il calcul l'angle charnière pour le second point de S, le compare avec la référence et conserve le plus grand des deux. Après avoir répété cette procédure pour l'ensemble des points de S, notre algorithme retourne l'angle charnière α constituant la borne inférieur tel que $\alpha < \theta$. La complexité en temps de notre algorithme est de $O(m\log(n))$ car il appelle m fois la fonction décrite en Figure Figure 1 dont la complexité est de $O(\log(n))$. La Figure 2 illustre notre algorithme. Comme montré dans la proposition suivante, la comparaison entre deux angles charnières est réalisée en temps constant, la complexité globale de notre algorithme n'est donc pas modifié.

Proposition 4 Soit α_1, α_2 deux angles charnières. Nous pouvons vérifier en temps constant et avec un calcul en entier si $\alpha_1 > \alpha_2$.

La preuve est similaire à celle de la proposition 3.

Data: Image numérique S, Angle Pythagoricien θ
Result: $\alpha(P,Q,K)$
var HA, HA_{temps} * angle charnière *\;
HA = Calcul de l'angle charnière(premier point de S, θ);
for $\forall \vec{p} \in S \backslash \{\vec{p}_1\}$ **do**
  HA_{temps} = Calcul de l'angle charnière($\vec{p}$, θ);
  if $HA < HA_{temps}$ **then**
    $HA = HA_{temps}$
  end
end
return HA
Function `Calculer un ange charnière pour une image numérique`

3.3. Rotation d'images numériques par angles charnières.

Data: image numérique S, angle charnière α
Result: image tournée S'
var HA : angle charnière;
for $\forall \vec{p} \in S$ **do**
  HA = Calcul de l'angle charnière($\vec{p}, \alpha$);
  déplacer $\vec{p}$ en $(K, \lfloor \lambda + \frac{1}{2}, \rfloor)$ ou $(\lfloor \lambda + \frac{1}{2} \rfloor, K)$, selon le signe de K et stocker dans S';
end
return S'
Function `Rotation discrète`

Dans la Figure 3, nous présentons un algorithme de rotation pour une image numérique. Il suppose que le centre de rotation se trouve sur l'origine. Pour l'ensemble des points, il appelle la fonction décrite dans la Figure 1 afin de trouver l'angle charnière correspondant, angle charnière qui désigne la nouvelle position du point. Soit n la plus grande coordonnée pour l'ensemble des points de S, on sais qu'il y a au plus $4n^2$ point dans S. Nous pouvons en conclure que la complexité est de $O(n^2 \log(n))$. Les résultats obtenue par la rotation discrète par angles charnières sont exactement les mêmes que ceux obtenus par RED. L'avantage de notre méthode est de ne pas recourir au calcul flottant, ce qui la désigne particulièrement aux processeurs à arithmétique entière.

4. Obtention d'une paire d'angles charnières à partir d'une paire d'images numériques

Dans cette section, nous montrons comment obtenir, à partir d'une paire d'images numériques, dont la seconde est une rotation de la première, l'encadrement par deux angles charnière de l'angle de rotation pour passer de l'une à l'autre. Soit $A = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_l\}$ et $B = \{\vec{q}_1, \vec{q}_2, ..., \vec{q}_l\}$ deux ensembles de points de la grille dont on connaît les correspondances tel que $\vec{p}_i$ corresponde à $\vec{q}_i$. Pour A et B donnés, on obtient une paire d'angles charnières α_1, α_2, tel que $\alpha_1 < \gamma < \alpha_2$ ou γ est un angle de rotation cohérent pour la correspondance des points de A et B. Dans la suite, nous supposerons que A est l'ensemble d'origine et B l'ensemble de points tournés par l'angle γ.

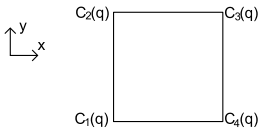


Figure 2: *La demi-grille $\mathscr{H}(\vec{q})$, à savoir le pixel autour du point $\vec{q}$ ainsi que ses quatre coins*

4.1. Définition du centre de rotation

Pour toute rotation, il est nécessaire de définir le centre de rotation. Dans cet article, nous prenons un des points de la grille appartenant à l'image numérique comme centre de rotation. En posant les centres de rotations en $\vec{p}_1$ et $\vec{q}_1$, on définit deux fonctions $\mathscr{T}_A$ et $\mathscr{T}_B$ tel que

$$\begin{aligned} \mathscr{T}_A(\vec{p}_i) &= \vec{p}_i - \vec{p}_1 \\ \mathscr{T}_B(\vec{q}_i) &= \vec{q}_i - \vec{q}_1 \end{aligned}$$

pour tout $\vec{p}_i \in A, \vec{q}_i \in B$, afin d'obtenir les centres de rotations sur l'origine après translations. Dans la suite, nous utiliserons les ensembles de points $A' = \{\mathscr{T}_A(\vec{p}_1), \mathscr{T}_A(\vec{p}_2), ..., \mathscr{T}_A(\vec{p}_l)\}$ et $B' = \{\mathscr{T}_B(\vec{q}_1), \mathscr{T}_B(\vec{q}_2), ..., \mathscr{T}_B(\vec{q}_1)\}$, à la place de A et B. Cependant pour simplifier la lecture, nous les noterons par $A = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_l\}$ et $B = \{\vec{q}_1, \vec{q}_2, ..., \vec{q}_l\}$.

4.2. calcul des angles charnières pour deux paires de points en correspondance

Dans cette sous-section, nous considérons le cas avec $A = \{\vec{p}_1, \vec{p}_2\}$ et $B = \{\vec{q}_1, \vec{q}_2\}$ où $\vec{p}_i = (P_i, Q_i)$ et $\vec{q}_i = (R_i, S_i)$. Définissons d'abord un cercle $\mathscr{C}(\vec{p}_2)$ de centre $\vec{p}_1$ passant par $\vec{p}_2$. Donc le rayon de $\mathscr{C}(\vec{p}_2)$ est $r = d(\vec{p}_1, \vec{p}_2)$ où $d(\vec{p}_1, \vec{p}_2)$ est la distance Euclidienne entre $\vec{p}_1$ et $\vec{p}_2$. Considérons la demi grille autour de $\vec{q}_2$ telle que

$$\begin{aligned} \mathscr{H}(\vec{q}_2) = \{(x,y) \in \mathscr{H} \ : \ & S_2 - \frac{1}{2} \leq y \leq S_2 + \frac{1}{2} \text{ quand } x = R_2 + \tfrac{1}{2}, \\ & R_2 - \frac{1}{2} \leq x \leq R_2 + \frac{1}{2} \text{ quand } y = S_2 + \tfrac{1}{2}\}. \end{aligned}$$

Posons $\vec{p}_1$ et $\vec{q}_1$ comme étant les centres de rotations, pour trouver la paire d'angles charnières, nous devons détecter les intersections entre $\mathscr{C}(\vec{p}_2)$ et $\mathscr{H}(\vec{q}_2)$. En d'autre termes, nous étudions les coins de $\mathscr{H}(\vec{q}_2)$ à l'intérieur de $\mathscr{C}(\vec{p}_2)$. Posons les quatre coins de $\mathscr{H}(\vec{q}_2)$ tel que $C_1(\vec{q}_2) = (R_2 - \frac{1}{2}, S_2 - \frac{1}{2})$, $C_2(\vec{q}_2) = (R_2 - \frac{1}{2}, S_2 + \frac{1}{2})$, $C_3(\vec{q}_2) = (R_2 + \frac{1}{2}, S_2 + \frac{1}{2})$, $C_4(\vec{q}_2) = (R_2 + \frac{1}{2}, S_2 - \frac{1}{2})$, comme montré dans la Figure 2. Nous définissons la fonction binaire $\mathscr{F}$ tel

que

$$\mathscr{F}(C_i(\vec{q}_2)) = \begin{cases} 0 & \text{si } C_i(\vec{q}_2) \text{ est en dehors de } \mathscr{C}(\vec{p}_2), \\ 1 & \text{sinon.} \end{cases}$$

Dans le but de calculer $\mathscr{F}(C_i(\vec{q}_2))$ uniquement avec des entiers, nous comparons chacune des équations de la forme $4((R_2 \pm \frac{1}{2})^2 + (S_2 \pm \frac{1}{2})^2)$ avec $4r^2$. Notons que nous pouvons supposer que les intersections entre $\mathscr{C}(\vec{p}_2)$ et $\mathscr{H}(\vec{q}_2)$ existent. Si il n'y a pas d'intersection entre $\mathscr{C}(\vec{p}_2)$ et $\mathscr{H}(\vec{q}_2)$, alors $\vec{p}_2$ et $\vec{q}_2$ ne sont pas en correspondance.

Proposition 5 Si deux points $\vec{p}_2$ et $\vec{q}_2$ sont en correspondance, un cercle $\mathscr{C}(\vec{p}_2)$ et le bord du pixel $\mathscr{H}(\vec{q}_2)$ possède deux intersections distinctes.

La preuve mathématique rigoureuse est omise dans cet article à cause de la limitation de pages. Par contre, les grandes lignes de la preuve sont donnée dans la suite. Nous distinguons les deux cas suivants.

Le premier cas qui se pose est celui ou $\mathscr{C}(\vec{p}_2)$ passe par l'un des coins de $\mathscr{H}(\vec{q}_2)$. Par définition des angles Pythagoriciens, tout les angles entre un point de la grille et le coin d'un pixel sont des angles Pythagoriciens, donc selon la Proposition 2 cela ne peut pas être un angle charnière.

Le second cas est celui ou l'intersection entre $\mathscr{C}(\vec{p}_2)$ et $\mathscr{H}(\vec{q}_2)$ est unique et sur une des arrêtes de $\mathscr{H}(\vec{q}_2)$. Ce cas pourrait arriver si une des coordonnées de $\vec{q}_2$ est zéro. Un arc de cercle centré sur l'origine ne peut croiser deux fois une droite parallèle à un des axes que si il coupe lui même l'autre axe. Donc si l'intersection est unique, elle doit être sur l'un des axes, ce qui implique que λ doit être nul. Cependant c'est impossible par définition des angles charnières.

A partir de la Proposition 5, on peut déduire qu'il y a quatre cas différents qui correspondent au différentes possibilité d'avoir 0,1,2 ou 3 coins à l'intérieur du cercle $\mathscr{C}(\vec{p}_2)$, comme illustré dans la Figure 3.

- Cas A : $\mathscr{C}(\vec{p}_2)$ ne contient aucun coin. Nous avons alors $\mathscr{F}(C_i(\vec{q}_2)) = 0$ pour tout $i = 1,2,3,4$, ce qui est similaire au second cas impossible. Ce cas ne peut arriver que lorsque $R_2 = 0$ ou $S_2 = 0$. Supposons que R_2 et S_2 soient strictement positif. Dans le premier quadrant, la coordonnée y (respectivement la coordonnée x) des points de $\mathscr{C}(\vec{p}_2)$ est strictement décroissante quand x (respectivement y) croit, donc il ne peut croiser deux fois une ligne parallèle à l'axe x (respectivement y). Par conséquent, si par exemple $S_2 = 0$, nous obtenons les deux angles charnières, symétriques autour de l'axe y, décrit par $\alpha(P_2, Q_2, R_2 - 1)$.
- Cas B : $\mathscr{C}(\vec{p}_2)$ contient un seul coin. Par exemple, si $C_1(\vec{q}_2)$ est dans le cercle, nous obtenons les deux angles charnières $\alpha_1(P_2, Q_2, R_2 - 1)$ et $\alpha_2(P_2, Q_2, -S_2 + 1)$.
- Cas C : $\mathscr{C}(\vec{p}_2)$ contient deux coins possèdant une coordonnée commune. Par exemple, si $C_1(\vec{q}_2)$ et $C_2(\vec{q}_2)$ sont dans le cercle, nous obtenons les deux angles charnières $\alpha_1(P_2, Q_2, -S_2 + 1)$ et $\alpha_2(P_2, Q_2, -S_2)$.
- Cas D : $\mathscr{C}(\vec{p}_2)$ contient trois coins. Par exemple, si $C_1(\vec{q}_2), C_2(\vec{q}_2)$ et $C_4(\vec{q}_2)$ sont à l'intérieur de $\mathscr{C}(\vec{q}_2)$, nous obtenons les deux angles charnières $\alpha_1(P_2, Q_2, R_2)$ et $\alpha_2(P_2, Q_2, -S_2)$.

La fonction principale de notre algorithme pour trouver les deux angles charnières, illustré dans la Figure 4, peut être divisée en trois parties. La première étape définit le centre de rotation sur $\vec{p}_1$ et $\vec{q}_1$ et applique les fonctions définies en 4.1. La seconde étape cherche quels sont les coins se trouvant à l'intérieur de $\mathscr{C}(\vec{q}_2)$ et conserve le résultat sous la forme d'un index. L'index est calculé comme la somme des index associés à chacun des coins tel que l'index du coin $C_i(\vec{q}_2)$ est égal à 2^i. Il est donc facile d'identifier quels coins se trouvent à l'intérieur de $C_i(\vec{q}_2)$ à partir de l'index. La troisième étape appelle la fonction illustré dans la Figure 5 qui renvois les angles charnières correspondant à l'index. Il existe quatorze valeurs possibles pour l'index. Notons que l'index ne peut jamais prendre les valeurs 5 ou 10. La valeur d'index 15 révèle une erreur car elle implique que tout les points se trouvent à l'intérieur de $\mathscr{C}(\vec{q}_2)$. Du fait que la valeur 0 corresponde au Cas A, nous devons vérifier si $\mathscr{H}(\vec{q}_2)$ possède vraiment deux intersections avec $\mathscr{C}(\vec{q}_2)$. Les autres valeurs de l'index, peuvent être classées par paires. Deux valeurs de l'index (d,e) appartiennent à la même paire si et seulement si $d + e = 15$. Deux valeurs d'une même paire désignent les même angles charnières. Du fait que cet algorithme travaille sur un nombre constant de points et que la quantité de calculs est indépendante des points, sa complexité en temps est constante en $O(1)$.

Data: Points
$\vec{p}_1(P_1, Q_1), \vec{p}_2(P_2, Q_2), \vec{q}_1(R_1, S_1), \vec{q}_2(R_2, S_2)$
Result: angles charnières $\alpha_1(P, Q, K_1), \alpha_2(P, Q, K_2)$
Var Entier $Rayon_1$, $Rayon_2$, $Index_{coin}$, K_1, K_2;
$\vec{p}_1 = \mathscr{T}_p(\vec{p}_1)$; $\vec{p}_2 = \mathscr{T}_p(\vec{p}_2)$; $\vec{q}_1 = \mathscr{T}_q(\vec{q}_1)$;
$\vec{q}_2 = \mathscr{T}_q(\vec{p}_2)$;
$Rayon_1 = 4 * (P_2^2 + Q_2^2)$; $Rayon_2 = 4 * (R_2^2 + S_2^2)$;
$Index_{coin} = \Sigma_{i=1}^{4} \mathscr{F}(C_i(\vec{q}_2)) * 2^{i-1}$;
(K_1, K_2) = Trouver la paire d'angles charnières
$(q_2(R_2, S_2), Index_{coin})$;
return $(\alpha_1(R_2, S_2, K_1), \alpha_2(R_2, S_2, K_2))$
Function `Trouver l'encadrement de l'angle`

S

4.3. Calcul des angles charnière de manière incrémentale

D'une manière générale, les ensembles de points A et B contiennent plus de deux points. C'est pourquoi dans cette section, nous étendons notre algorithme présenté dans la section précédente pour toute paires d'ensembles de points A et B qui contiennent m points avec $m > 2$. Un nouvel algorithme illustré dans la Figure 6 effectue les calculs pour l'ensemble des points de manière incrémentale. Cet algorithme est divisé en deux parties. La première partie consiste à initialiser

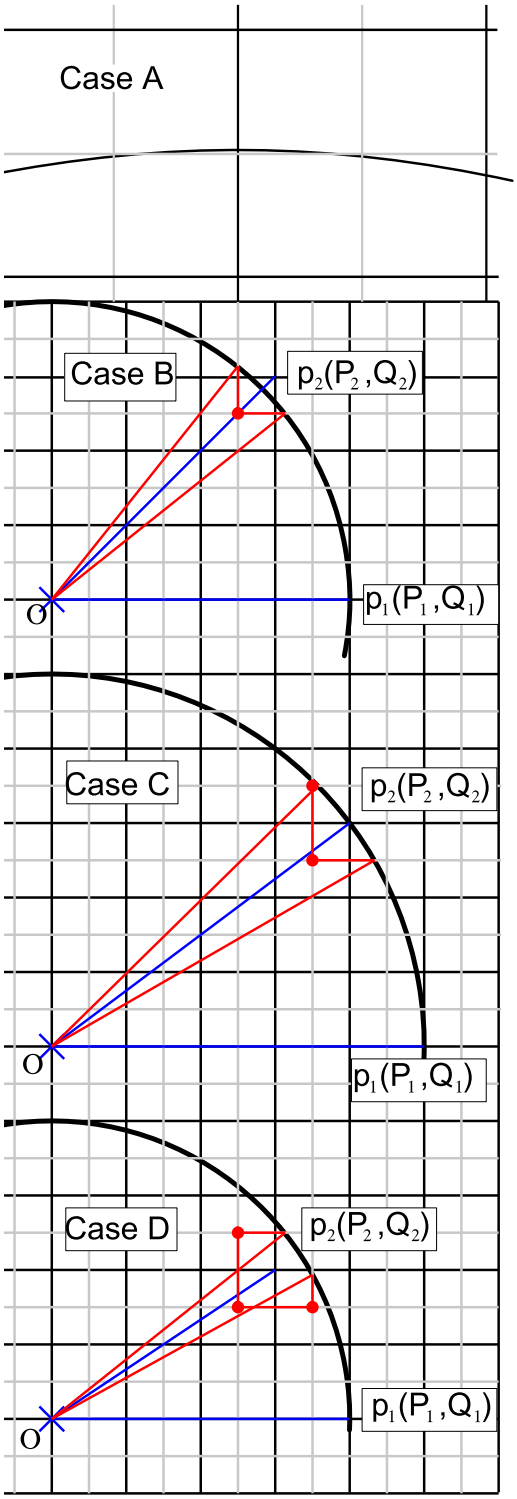


Figure 3: *Illustration des cas 1,2,3 et 4.*

Data: point $\vec{q}_2(R_2,S_2)$, index $Index_{coin}$
Result: Deux entiers pour définir les angles charnières
if $Index_{coin} = 0$ **then**
  if $|\sqrt{Rayon_1} - \sqrt{Rayon_2}| > 2$ **then**
    return *erreur point non compatible*
  end
  if $R_2 = 0$ **then**
    return $-S_2+1, -S_2+1$
  else
    return R_2-1, R_2-1
  end
end
if $Index_{coin} = 1$ *or* $Index_{coin} = 14$ **then**
  return $R_2-1, -S_2+1$
end
if $Index_{coin} = 2$ *or* $Index_{coin} = 13$ **then**
  return $R_2-1, -S_2$
end
if $Index_{coin} = 4$ *or* $Index_{coin} = 11$ **then**
  return $R_2, -S_2$
end
if $Index_{coin} = 8$ *or* $Index_{coin} = 7$ **then**
  return $R_2-1, -S_2+1$
end
if $Index_{coin} = 3$ *or* $Index_{coin} = 12$ **then**
  return $-R_2+1, -S_2$
end
if $Index_{coin} = 6$ *or* $Index_{coin} = 9$ **then**
  return R_2-1, R_2
end
if $Index_{coin} = 15$ **then**
  return *erreur point non compatible*
end
Function `Trouver la paire d'angles charnières`

la première paire d'angles charnières en appelant la fonction de la Figure 5 sur les deux paires de points $(\vec{p}_i, \vec{q}_i)$ pour $i = 1,2$. Cette paire d'angles charnières sert de référence pour la suite de l'algorithme. La seconde partie boucle sur l'ensemble des paires de points afin de trouver de nouveaux angles charnières qui améliorent l'encadrement de γ. Pour toutes les paires de points, on appelle la fonction de la Figure 5 et on compare le résultat avec la référence afin de la modifier si nécessaire.

La complexité en temps de cet algorithme est $O(m)$. Comme expliqué dans la sous-section précédente, la fonction de la Figure 5 est calculé en un temps $O(1)$. De plus, comme expliqué dans la Section 3, nous pouvons comparer deux angles charnières en un temps constant $O(1)$. Nous pouvons en conclure que l'ensemble des calculs de cet algorithme pour m points prennent un temps $m \times (O(1) + O(1)) = O(m)$.

Data: Ensemble de points
$A = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_n\}, B = \{\vec{q}_1, \vec{q}_2, ..., \vec{q}_n\}$
Result: une paire d'angle charnière α_1, α_2
Var angle charnière $\alpha_1, \alpha_2, Temps\alpha_1, Temps\alpha_2$;
(α_1, α_2) = Trouver l'encadrement de l'angle $(\vec{p}_1, \vec{p}_2, \vec{q}_1, \vec{q}_2)$;
for *i = 3; i< n, i++* **do**
  $(Temps\alpha_1, Temps\alpha_2)$ = Trouver l'encadrement de l'angle $(\vec{p}_1, \vec{p}_i, \vec{q}_1, \vec{q}_i)$;
  if $Temps\alpha_1 > \alpha_1$ **then**
    $\alpha_1 = Temps\alpha_1$;
  end
  if $Temps\alpha_2 < \alpha_2$ **then**
    $\alpha_2 = Temps\alpha_2$;
  end
  if $\alpha_2 < \alpha_1$ **then**
    Retourner Erreur
  end
end
Retourner (α_1, α_2);

Function `Calcul des angles charnière incrémentale`

5. Conclusion

Dans cet article, nous avons montré comment obtenir un angle charnière directement inférieur à un angle Euclidien donné. Nous avons présenté ensuite un algorithme efficace pour effectuer une rotation discrète à partir de l'angle charnière précédemment trouvé, et ceux avec le même résultat que l'angle Euclidien. Nous avons enfin montrés comment obtenir, à partir d'une paire d'images numériques, le triplet d'entiers qui définit la relation géométrique entre ces deux images.

References

[And92] ANDRES E.: *Incremental and transitive discretized rotations*. PhD thesis, Université Louis Pasteur Strasbourg, 1992.

[And96] ANDRES E.: The quasi-shear rotation. 307–314.

[Ang88] ANGLIN W. S.: Using pythagoricien triangles to approximate angles. *American Mathematical Monthly*, 95 (1988), 540–541.

[NR06] NOUVEL B., RÉMILA E.: Incremental and transitive discretized rotations. *IWCIA* (2006), 199–213.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

L'enseignement en informatique graphique et le processus de Bologne

Abstract

La déclaration de Bologne a modifié les conditions d'enseignement dans le supérieur. Il est important pour nous, qui sommes souvent enseignants-chercheurs, de regarder quelles conséquences elle entraîne en particulier pour les enseignements en Informatique Graphique. Un premier workshop s'est tenu à Vienne en 2006, à l'initiative de l'ACM et de EUROGRAPHICS, pour traiter de cette question et des nouveaux cursus à mettre en oeuvre. Nous détaillerons ici les principales étapes de ce processus, ses résultats et les, nombreuses, questions ouvertes.

1. Le contexte

L'unification européenne ne peut pas aller sans un minimum d'homogénéisation des politiques universitaires, tant au niveau recherche qu'au niveau enseignement. Si l'harmonisation de la recherche se fait essentiellement au moyen d'appels à projets et, donc, d'incitations financières fortes, il n'en va pas de même avec l'enseignement supérieur. Les conditions d'exercice ne sont pas les mêmes à l'intérieur de l'Union Européenne, les cursus ne suivent pas les mêmes axes et durées. Il semblait donc important de tenter un mouvement de rapprochement.

En 1999 fut signée par 29 ministres européens de l'enseignement supérieur une déclaration [Uni99] visant à créer les conditions d'une homogénéisation entre les différentes pratiques d'enseignement supérieur. Cette réforme n'est pas un texte contraignant pour les pays de l'UE. Il est conçu, au contraire, comme un moyen pour les universités européennes ainsi que pour des universités extérieures à l'Europe, d'assurer une convergence. Il y a, au premier octobre 2007, 46 pays adhérents, c'est-à-dire bien plus que de pays membres de l'UE.

1.1. La déclaration de Bologne

Il faut ici reprendre le texte de cette déclaration pour en mesurer les ambitions et les limites. D'une part le but principal est de permettre les études transfrontalières, de faire que davantage d'étudiants suivent une partie de leurs études supérieures à l'étranger, d'autre part d'attirer les étudiants de pays tiers, en particulier en améliorant la lisibilité de l'offre de formation européenne. Ces ambitions s'accompagnent de contraintes comme celle de systématiser les ECTS, l'uniformisation des cursus d'étude et l'amélioration de la "qualité" de l'enseignement supérieur. Étudions brièvement ces trois points.

1.1.1. ECTS

Le système ECTS (european credit transfer system) donne une certaine valeur ECTS à chaque enseignement capitalisable en fonction du nombre d'heures de travail qu'il est censé demander aux étudiants. Une année d'étude donne droit à 60 points ECTS. Sur une base de 1800 heures de travail annuel, chaque point correspond à la validation de 30 heures de travail effectif. Bien sûr ces valeurs sont théoriques et les experts européens ne sont pas entrés dans des considérations aussi appliquées, personne n'arrivant à se mettre d'accord sur combien d'heures de travail est nécessaire pour la réussite d'un diplôme (sans parler des concours d'entrée en école d'ingénieurs en France).

Pour un étudiant donné, ce système permet de suivre une partie de son cursus à l'étranger, ou même dans une autre université et de voir les cours, suivis et validés, reconnus à son retour. Il obtient alors les points ECTS correspondants. Il a passé un semestre à l'étranger, a obtenu 30 points, il continue son diplôme en validant un semestre complet de 30 points comme s'il avait fait cette partie de ses études dans son université d'origine.

Le système ECTS est utilisé depuis un certain nombre d'années et ses possibilités et limites sont bien connues : 5 points ECTS ne valent pas 5 points ECTS ! En effet les points doivent être associés à un niveau donné pour être pertinents. Imaginons qu'un étudiant de master

informatique demande à valider des points ECTS obtenus en première année de licence d'une autre université (Introduction à l'informatique 5 points, Introduction aux calculs logiques 5 points…), il ne peut évidemment pas les utiliser. Donc les points doivent être indexés de l'année d'étude. Plus généralement ils doivent faire l'objet d'un contrat pédagogique préalable. Dans le cadre des relations internationales c'est le plus souvent la règle : l'étudiant signe avec son encadrant un accord sur les cours qu'il va suivre dans l'université partenaire avant même de partir. Cet accord doit être généralisé au sein même de la formation ou de l'université. Des étudiants en Informatique Graphique pourraient renoncer à un cours de système d'exploitation et lui préférer un cours d'écriture de scénario, bien utile pour monter un projet d'animation.

1.1.2. Les trois grades

Le système mis en place privilégie deux niveaux d'étude dans le supérieur. Le master doit être obtenu à l'issue de cinq années d'études. Le niveau inférieur est le "bachelor" (ici la licence) qui est obtenu après au moins trois années d'études. Si plusieurs pays ont choisi que le diplôme intermédiaire serait obtenu après trois années, d'autres, comme l'Espagne, choisissent un bachelor au bout de quatre années. Il a fallu attendre quelques années après la déclaration de Bologne pour que le doctorat soit mentionné, il se passe trois ans après le master.

Le bachelor correspond donc à 180 ou 240 points ECTS et le master à 60 ou 120 points de plus, mais donc à un total de 300 points ECTS.

Un point important de la déclaration, souvent omis, est que ces diplômes doivent correspondre au marché de l'emploi. "Les diplômes délivrés au terme du premier cursus correspondront à un niveau de qualification approprié pour lŠinsertion sur le marché du travail européen" [Uni99]. Il est donc important de mettre en relation le diplôme et la quête d'emploi dans le secteur au niveau concerné. Nous verrons plus loin une conséquence de cet aspect.

1.1.3. L'assurance qualité

Il est dit que cette réforme doit s'accompagner d'une garantie de qualité des études sans que ceci soit matérialisé ou éclairé. La mise en place de l'AERES, en remplacement des CNE et comité d'experts du ministère est sans doute la forme de l'adoption par la France de cette assurance qualité.

2. Premières conséquences de la réforme

Les conséquences de la réforme ont donné lieu à une évaluation par la commission européenne. Nous ne détaillerons pas ici la totalité d'un document d'une centaine de pages [RT] mais nous devons en tirer quelques conclusions.

2.1. Homogénéisation

L'homogénéisation des études a été plus que partielle. Si certains pays ont tenu a organiser l'ensemble des études selon le mode issu de Bologne, cas de la France, tous les pays ne convergent pas vers des durées d'études communes. Par ailleurs les modes de passage au système de Bologne est très diversifié, certains pays laissent les universités quasiment libres de faire ce qu'elles veulent tandis que d'autres systèmes sont beaucoup plus contraints. Il s'est agit, en France, d'une réforme d'envergure qui couvrait d'autres points que ceux de la déclaration de Bologne, comme, par exemple, la quasi uniformisation des dénominations des offres de formation et l'obligation de faire entrer toute formation dans un arbre "domaine/mention/spécialité/parcours". Le tout accompagné d'une directive visant à impulser des cursus pluridisciplinaires (avec les systèmes de disciplines majeure et mineure concourant à un même diplôme).

Un autre problème est venu de la demande de professionnalisation des diplômes. Elle a suscité les critiques : "Dans de nombreuses universités, les professeurs et, à un moindre degré, les doyens, parfois aussi la direction, expriment encore des doutes profonds sur la possibilité de proposer un diplôme sŠobtenant en trois ans seulement qui soit signifiant tant au niveau universitaire que sur le marché du travail" Elle a conduit à deux attitudes divergentes : soit en concentrant en trois ans tout ce qui s'enseignait antérieurement en cinq ans, dans un diplôme connu pour ses débouchés professionnels, soit en réduisant la licence à un diplôme professionnalisant, type licence-pro en France. Dans les deux cas l'enseignement supérieur est dénaturé.

2.2. Multiplication des cursus

En réalité, dans beaucoup de cas, contre l'avis des ministres concernés, cette réforme s'est accompagnée d'une multiplication des diplômes. Il est vrai que pour n diplômes monodisciplinaires on peut faire environ n^2 diplômes bidisciplinaires ! Les auteurs du rapport ne font pas ce ratio et remarquent : "on constate souvent une tendance à créer trop de nouvelles formations, parce que ń tous les professeurs veulent avoir la leur ż".

Les ministres tentent maintenant de réduire drastiquement ce nombre de diplômes et nous devrions voir bientôt de nombreux diplômes refusés par les instances nationales.

2.3. Visibilité et attrait des études

Le problème sous-jacent est que l'attrait les études en Europe n'a pas augmenté contrairement aux voeux des signataires. Le nombre d'étudiants venus de pays tiers pour suivre des études en Europe ne semble pas progresser, en particulier le nombre des "meilleurs d'entre eux". C'est pour cela que la commission a mis en place les masters "Erasmus mundus". Ces masters de très bon niveau doivent

être montés et suivis par des établissements d'enseignement supérieur d'au moins trois pays. Chaque étudiant doit suivre des études dans au moins deux pays et doit avoir à pratiquer au moins deux langues européennes. À ces masters sont associés des moyens considérables ainsi que des bourses significatives pour les enseignants et les étudiants des pays tiers (21 000 euros/an). Malgré cela de nombreux étudiants bénéficiaires des bourses renoncent chaque année, après avoir obtenu, souvent, des offres plus intéressantes...

C'est néanmoins pour essayer d'améliorer la visibilité des enseignements supérieurs en informatique graphique que l'ACM et EUROGRAPHICS ont décidé de monter un Workshop en 2006, à Vienne, sur cette questionă: "Defining an International Curriculum in Computer Graphics" Le principe que nous avions choisi était de réunir des enseignants-chercheurs de notre domaine pour tenter de voir comment nous pouvions définir un tel cursus à partir des contraintes de la déclaration de Bologne.

3. Le Workshop

Le workshop cherchait donc à définir un cursus commun d'informatique graphique (computer graphics) destiné à des publics d'étudiants en sciences (les cursus arts et informatique nombreux aux USA devant faire l'objet d'un travail spécifique). Il devait se dérouler sur une journée.

3.1. Préparation

Nous avions retenu le principe d'un appel à contribution puis d'une sélection rigoureuse des participants auxquels devaient être associés les organisateurs ainsi que les membres du Comité Education de EUROGRAPHICS et de l'ACM-SIGGRAPH. Le lieu et la dates coïncidaient avec le samedi suivant le conférence EUROGRAPHICS à Vienne en septembre 2006. L'organisation de la conférence a donc beaucoup fait pour le succès de ce workshop, de même que la NSF et l'ACM qui ont pris en charge une partie des frais de certains participants. De nombreuses contributions ont été reçues et une quinzaine d'entre elles a été sélectionnée. Au total 23 participants ont contribué à ce travail. La liste complète sera trouvée dans le texte du rapport [BCFH06].

3.2. Organisation

Dans un premier temps nous avions décidé de traiter les questions en deux temps : le ou les cours d'introduction dans le premier cycle puis les cours de spécialisation.

Cette répartition s'est heurtée à la situation de ne pas savoir quelle durée, en nombre d'années dure le premier cycle. Nous avons résolu cette question en supposant que, de toutes façons, il y aurait trois années d'abord, que nous allions travailler sur ce qu'il fallait proposer au cours de ses trois premières années, et qu'il y aurait deux années ensuite, à traiter séparément. Que cette séparation ne dépendait pas de savoir si un diplôme pouvait être obtenu au bout de trois ou quatre années d'études.

Nous avons travaillé en assemblée plénière ou par groupe, alternativement. Les remarques et les propositions furent nombreuses et il ne fut pas facile de converger vers un résultat commun.

3.3. Résultats

Nous avons suivi la recommandation d'un précédent groupe de travail qui concluait au besoin d'un cours d'introduction à l'informatique graphique [CHLS04] en y ajoutant l'idée que ce cours devait être aussi un cours sur la communication visuelle.

N'entrons pas dans les détails des prérequis, qui vous paraîtront évidents, comme des bases en mathématique, en géométrie, ou même en traitement d'images.

Il n'a pas semblé possible d'imposer que toute licence informatique comprenne plus d'un cours d'informatique graphique, même si chacun souhaite le faire dans son propre cursus. Nous nous sommes alors concentrés sur une séparation en cours "de base" et cours "avancés".
Les cours de base :

- Rendu I
- Modélisation I
- Animation I
- Visualisation I
- Programmation GPU
- Interaction

Les cours de avancés :

- Rendu II
- Modélisation II
- Animation II
- Visualisation II
- Rendus temps-réels

Au cours du workshop et dans le compte-rendu, ces axes sont détaillés et il est plus facile de différencier les niveaux I et II de chaque matière.

4. Questions ouvertes

Au cours de la journée, de même qu'au cours des discussions qui l'a suivie, de nombreux problèmes ont été ouverts. D'abord se pose la question de savoir si nous entendons tous la même chose dans ces définitions. Il n'est pas indifférent de savoir s'il s'agit d'écrire des programmes qui vont faire un travail ou s'il s'agit d'utiliser des logiciels qui le feront... Par exemple, Marc Bailey nous prie d'inclure la programmation GPU dès l'abord mais la met en place grâce à *glman* qui éloigne l'étudiant de la programmation [BC07]. On peut, de même, présenter un sujet de façon magistrale ou le faire travailler par les étudiants jusqu'à ce qu'ils obtiennent un ré-

sultat tangible, cela ne prendra pas le même temps et ne conduira pas aux mêmes compétences. On pourra lire à ce sujet l'article de Steffi Beckhaus et al. sur la pratique, jusqu'à l'enseignement d'une compétence [BB06].

Nous n'avons donc pas beaucoup avancé dans le rapprochement, sous le même intitulé peuvent se cacher des cours très différents. Une proposition serait que chaque cours soit doté d'un ouvrage de référence, d'un manuel de cours, avec les exercices correspondants, de façon à réellement unifier nos pratiques. Ces manuels pourraient utiliser CGEMS pour être diffusés [EUR].

Enfin, il n'est pas certain que tous les enseignants souhaiteraient un cursus unifié de a à z. Nombreuses sont les formations disposant de spécificités et nul ne songerait sérieusement à les gommer. Il s'agirait plutôt, alors, de décrire des éléments de cursus et de créer des spécialités une fois ce programme commun atteint.

Au cours du programme enseignement de EUROGRAPHICS 2007, à Prague, la question de comment se passait l'application de la déclaration de Bologne. Trois exemples européens ont été présentés (Barcelone UAB, Paris 8, Hambourg) et longuement discutés. Visiblement l'uniformité est loin de nous.

5. Conclusion

La mise en place du processus de Bologne s'accompagne de nombreux bouleversements de nos pratiques. Le workshop de 2006 tentait de nous aider à passer cette difficulté. Ses résultats sont loin d'être suffisants. Avancer vers un cursus commun n'est peut-être pas un but satisfaisant mais il faudrait au moins savoir de quoi nous parlons et ce que savent réellement les étudiants qui réussissent nos formations et celles de nos confrères.

Au cours de la réunion du comité *éducation* d'EUROGRAPHICS (Prague, septembre 2007) il a été demandé à tous les chapitres de transmettre notre souhait de voir plus de soumissions à CGEMS [EUR]. Nous avons également souhaité un nouveau workshop sur les conséquences de la déclaration de Bologne, il devrait avoir lieu cet été à Barcelone. Nous souhaitons une forte participation à ce workshop.

References

[BB06] BECKHAUS S., BLOM K.: Teaching, exploring, learning - developing tutorials for in-class teaching and self-learning. In *Eurographics 2006, Education Paper* (September 2006), Association T. E., (Ed.), pp. 13–21.

[BC07] BAILEY M., CUNNINGHAM S.: A hands-on environment for teaching gpu programming. *SIGCSE Bull. 39*, 1 (2007), 254–258.

[BCFH06] BOURDIN J.-J., CUNNINGHAM S., FAIRÉN M., HANSMANN W.: Defining an international curriculum in computer graphics, 2006. http://education.siggraph.org/conferences/eurographics/2006/cge2006/cge-06-report-pdf.

[CHLS04] CUNNINGHAM S., HANSMANN W., LAXER C., SHI J.: The beginning computer graphics course in computer science. http://education.siggraph.org/conferences/eurographics/cge-04/Rep2004CGEworkshop.pdf, 2004.

[EUR] Cgems. http://cgems.inesc.pt/.

[RT] REICHERT S., TAUCH C.: Trends iv : Etat de la mise en ouvre des reformes de bologne par les universites europeennes. European University Association. http://www.eua.be/index.php?id=128.

[Uni99] La déclaration de bologne, 1999. http://www.education.gouv.fr/realisations/education/superieur/bologne.htm.

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Une approche multirésolution lagrangienne pour la simulation de vagues déferlantes

Darles E., Crespin B., Ghazanfarpour D.

XLIM - Université de Limoges

Abstract

Dans cet article, nous présentons une nouvelle méthode afin de simuler le déferlement plongeant appliqué à de larges scènes océaniques. De nouvelles équations afin de simuler ce phénomène sont présentées. La génération des sprays est effectuée par une transition d'état de certaines particules d'eau en corrélation avec la quantité de mouvement et de turbulence produite par le déferlement. L'utilisation d'une approche lagrangienne associée à un schéma de multirésolution nous permet d'accélérer les calculs et de simuler cet effet sur de grandes étendues d'eau.

1. Introduction

De nombreux travaux en mathématiques et physiques appliquées ont pour sujet la représentation des vagues déferlantes et de la dynamique littorale, rendue très complexe en raison des interactions entre la mer, le continent, l'atmosphère et parfois les fleuves et les infrastructures humaines. De nombreux phénomènes entrent en jeu (marées, vent, courants), et on rencontre une grande diversité d'échelles d'espace et de temps auxquelles apparaissent les différentes forces régissant la circulation littorale. Nous nous intéressons ici principalement au phénomène des vagues déferlantes, qui apparaissent en eau peu profonde près du rivage. Ces vagues sont caractérisées individuellement par un mouvement relativement simple (levée, déferlement plus ou moins important, retrait puis stabilisation), mais présentent tout de même un challenge pour les simulations numériques.

En infographie, les scènes océaniques sont nombreuses et si de nombreuses méthodes ont pour objet la représentation des fluides en général, paradoxalement peu d'entre elles traitent spécifiquement du déferlement. Nous présentons ici une solution efficace afin de représenter ce phénomène à grande échelle, en intégrant également la modélisation et le rendu des sprays dûs au déferlement. Après avoir survolé les différentes méthodes envisageables, nous présentons le modèle SPH choisi pour discrétiser le fluide, puis différentes méthodes permettant d'optimiser la simulation dans le cas spécifique des vagues déferlantes : nous nous intéressons notamment à la génération de vagues non limitée dans le temps, et à la représentation multirésolution du phénomène.

2. Travaux Antérieurs

Les premiers travaux concernant la représentation de l'eau dans des scènes océaniques remontent aux modèles paramétriques [FR86, DP86], étendus pour permettre la représentation réaliste de trains de vagues en eau profonde [TDG00, CGG01]. Les vagues sont définies par une équation paramétrique décrivant le trajet des particules d'eau à la surface au cours du temps. Sous certaines conditions il est possible d'obtenir une forme réaliste de vagues déferlantes [JBS03], cependant ce type d'approche ne permet pas de simuler la dynamique du fluide. A l'inverse, les méthodes eulériennes proposent une résolution du système régissant le mouvement du fluide. Cette résolution peut se baser par exemple sur une grille de hauteurs 2D incorporant les effets de pression [KM90] ; la troisième dimension est obtenue à l'aide d'une approche paramétrique. Différents types de vagues déferlantes peuvent être représentés [MMS04] en permettant le contrôle de la géométrie de départ de la vague. Une approche similaire est utilisée dans [TMFSG07] pour simuler le déferlement en temps réel. La résolution 3D du phénomène [LGF04, IGLF06, LSSF06] permet d'intégrer des effets réalistes de splashs, mais au prix d'une grille 3D très fine, notamment dans le cas de scènes océaniques.

Les approches semi-langrangiennes combinent une grille eulérienne avec un système de particules, et permettent donc

de simuler correctement des effets allant de l'écoulement dans un verre jusqu'aux splashs dûs au déferlement, tout en conservant la masse totale du système [OH95, FF01, EMF02, TFK*03]. Cette approche a été utilisée récemment pour modéliser les bulles et les sprays dûs au déferlement à l'aide de systèmes de particules [KCC*06], générés à la volée en quantifiant le caractère dissipatif et turbulent du fluide. Cette approche a également été utilisée pour la résolution des équations de Saint-Venant en incorporant une fonction de turbulence [TRS06], grâce à un système de particules représentant les sprays et l'écume engendrés par celle-ci. Les multifluides peuvent être simulés avec une grille adaptative, raffinée lorsque la cellule est proche de l'interface [LSSF06].

Enfin, la dernière approche consiste à simuler la dynamique du fluide de manière purement lagrangienne, c'est à dire en se basant uniquement sur un système de particules, généralement en suivant le modèle des SPH [Mon92, DC96]. Müller *et al.* [MCG03] ont notamment proposé une formulation adaptée au cas de l'eau, permettant de calculer de façon correcte la densité d'une particule par rapport à ses voisines ; afin de réduire la compressibilité, cette densité peut être filtrée par un schéma de relaxation [CBP05]. L'un des avantages d'un système de particules est que certaines peuvent être contrôlées (par l'utilisateur ou par un autre type de simulation) par la modification du champ des vitesses [MSKG05], ce qui pourrait convenir à la simulation du déferlement. D'autres approches se basent sur le modèle MPS, notamment pour représenter l'écoulement et les interactions incompressibles d'un système multifluides [PTB*03]; ce modèle a été employé également pour la simulation du déferlement [QYC*06] par une série de "profils" 2D qui sont ensuite joints pour reconstruire une vague déferlante.

Le temps de calcul important reste le problème principal des méthodes précédentes - de l'ordre d'une dizaine de minutes par image dans le cas de vagues déferlantes. Dans ce contexte, les approches multirésolution permettent de réduire la complexité des données. On a déjà vu que certains schémas de résolution eulériens [LGF04] utilisent une grille raffinée dans les zones proches de l'interface air-eau ou en fonction d'autres critères, par exemple l'impact visuel de la zone de fluide. Les particules sont un modèle également bien adapté à ce principe [DC99, APKG07] : la taille des particules varie selon un schéma "fusion / division" permettant de générer plus de particules dans les zones d'importance. Le problème réside alors dans l'interaction entre des particules de tailles différentes, qui peut être résolu en modifiant les calculs de pression et de viscosité. Notre méthode se situe dans la continuité de ces travaux, en présentant une méthode multirésolution adaptée au cas des vagues déferlantes.

3. Approche SPH

3.1. Concept

Les "Smooth Particles Hydrodynamics" sont une discrétisation lagrangienne permettant la résolution des équations de Navier-Stokes. Pour une particule donnée, le calcul d'une quantité scalaire A_i associée à une particule i s'effectue par interpolation des particules se trouvant à son voisinage dans un rayon d'interaction noté h :

$$A_i = \sum_j m_j \frac{A_j}{\rho_j} W(r_i - r_j, h) \quad (1)$$

où m_j désigne la masse de la particule j, A_j son champ scalaire, ρ_j sa densité et $r_i - r_j$ la distance entre les deux particules. W est un noyau d'interpolation normalisé à support compact et fini :

$$W(r,h) = W(r,-h)$$
$$\int W(r)dr = 1$$

Le gradient et le laplacien d'une quantité A_i peuvent être calculés respectivement par :

$$\nabla A_i = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(r_i - r_j, h)$$
$$\nabla^2 A_i = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(r_i - r_j, h)$$

3.2. Résolution des équations de Navier-Stokes

Sous forme eulérienne, les équations Navier-Stokes s'expriment à l'aide du système :

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho v) = 0$$
$$\frac{\partial v}{\partial t} = \frac{1}{\rho}(-\nabla p + \mu \nabla^2 v - \rho v \nabla v + \rho G)$$

où ρ désigne la densité de fluide présent dans la cellule considérée, v sa vitesse, p sa pression, μ sa viscosité et G le vecteur gravitationnel. * La première équation représente la conservation de la densité : au cours du temps, la densité de fluide présente dans le système global est constante. La seconde représente la conservation des moments et met en jeu différents termes : $-\nabla p$ représente les effets de pression, $\mu \nabla^2 v$ les forces de viscosité, $\rho v \nabla v$ les effets d'advection (c'est-à-dire la diffusion de fluide d'une cellule à une autre) et ρG désigne les forces extérieures.

Dans une approche lagrangienne, ce système d'équation se simplifie. En considérant la masse des particules comme constante au cours du temps, nous pouvons omettre première équation ; dans la seconde, le terme d'advection non linéaire peut être également omis. On obtient alors une seule équation de type liméaire :

$$\frac{\partial v}{\partial t} = \frac{1}{\rho}(-\nabla p + \mu \nabla^2 v + \rho G)$$

Notons $f = -\nabla p + \mu \nabla^2 v + \rho G$ désignant l'ensemble des forces agissant sur une particule. Son accélération peut être

alors calculée par :

$$a_i = \frac{dv_i}{dt} = \frac{1}{\rho_i}(f_i^{pres} + f_i^{visc} + f_i^{ext})$$

où f_i^{press}(resp. f_i^{visc} et f_i^{ext}) désigne la force de pression (resp. viscosité et extérieures) et ρ_i la densité associées à une particule.

Dans l'approche SPH, la densité peut être calculée à l'aide de l'équation 1, et la pression à l'aide de loi des gaz parfaits :

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$
$$p = k(\rho - \rho_0)$$

où k désigne la solubilité du gaz caractéristique du fluide considéré et ρ_0 sa densité au repos.

Comme cela est suggéré dans [MCG03], les forces de pression et de viscosité sont alors obtenues grâce aux noyaux W^{spiky} et $W^{viscosity}$ par :

$$f_i^{press} = -\sum m_j \frac{p_i + p_j}{2\rho_j} \nabla W^{spiky}(r_i - r_j, h) \quad (2)$$

$$f_i^{visc} = \mu \sum m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W^{viscosity}(r_i - r_j, h) \quad (3)$$

La gravité et la tension de surface sont définies par :

$$f_i^{surface} = -\sigma \nabla^2 c_i^s \frac{n_i}{|n_i|} \quad (4)$$

$$c_i^s = \sum \frac{m_j}{\rho_j} W(r_i - r_j, h) \quad (5)$$

où σ représente le tenseur d'effort caractéristique du fluide considéré, c_i^s la fonction de couleur et n_i la normale à la particule définie par $n_i = \nabla c_i^s$. Cette force est calculée si $|n_i|$ est supérieur à un certain seuil.

Enfin, le rayon d'interaction h peut s'exprimer en fonction des propriétés du fluide et de son volume :

$$h = \sqrt[3]{\frac{3Vx}{4\pi n}} \quad (6)$$

avec V le volume de fluide total, n le nombre de particules présent dans la simulation et x le taux de particules maximal de particules pouvant échanger avec une particule quelconque, dépendant des propriétés du fluide considéré. Une valeur de x proche de 20 permet de simuler correctement l'eau tout en réduisant les effets de compressibilité dûs à l'approche SPH.

4. Simulation de vagues déferlantes

4.1. Méthode par impulsion finie

Le déferlement d'une vague est effectué en plusieurs phases. Dans un premier temps, on observe une levée et une propagation d'une onde solitaire. Ensuite, la vague plonge, générant sprays et écume s'accompagnant d'un phénomène de turbulence. Enfin la surface de l'océan se stabilise et on observe une phase de retrait du fluide dû à la levée et à la propagation d'une nouvelle vague déferlante (voir Fig. 1).

Ce phénomène complexe a été longtemps étudié en océanographie. Le modèle le plus employé est celui du soliton [RO98], permettant de décrire une vague déferlante comme une onde solitaire se propageant dans un milieu non linéaire et dispersif. Les méthodes existantes [EMF02, MMS04] initialisent les conditions initiales de vélocité du fluide à l'aide de modèles paramétriques lui permettant ainsi de déferler.

Cependant, après un certain temps de simulation, le fluide n'a plus assez de vitesse pour déferler à nouveau. Pour permettre un nouveau déferlement, il faudrait alors réinitialiser la vélocité du système, ce qui provoquerait une divergence numérique et le rendrait instable. Ce type de méthode induit donc une limitation temporelle de la simulation. Pour pallier à cette limitation, nous proposons d'employer un système de forces d'impulsions basées sur la solution du soliton. A chaque instant une force d'impulsion, représentée comme une force extérieure, est appliquée à l'ensemble du fluide.

La vélocité initiale en 2D de chaque particule s'exprime par :

$$u = \sqrt{gd}\frac{H}{d} sech^2(\sqrt{\tfrac{3H}{4d^3}}(x - ct))$$
$$v = \sqrt{3gd}(\tfrac{H}{d})^{3/2}\tfrac{y}{d} sech^2(\sqrt{\tfrac{3H}{4d^3}}(x - ct))\tanh(\sqrt{\tfrac{3H}{4d^3}}(x - ct))$$

avec H la hauteur du soliton, c sa célérité, d la profondeur de la particule par rapport au fond et t son temps fictif, c'est-à-dire le temps propre du soliton indépendant du temps de la simulation.

Ce modèle, étendu au cas 3D, inclut un paramètre ϕ de déphasage du soliton, ainsi qu'un angle θ de propagation par rapport à l'axe de déferlement permettant d'obtenir un résultat asymétrique :

$$u' = \sqrt{gd}\frac{H}{d} sech^2(\sqrt{\tfrac{3H}{4d^3}}(x - ct + \phi))$$
$$v' = \sqrt{3gd}(\tfrac{H}{d})^{3/2}\tfrac{y}{d} sech^2(\sqrt{\tfrac{3H}{4d^3}}(x - ct + \phi))$$
$$w' = \sqrt{3gd}(\tfrac{H}{d})^{3/2}\tfrac{y}{d} sech^2(\sqrt{\tfrac{3H}{4d^3}}(z\cos\theta - ct + \phi))$$

A chaque instant, une force d'impulsion $F^{soliton}$ de coordonnées u',v' et w' est appliquée à l'ensemble des particules sous forme de force extérieure.

4.2. Modèle d'impulsions multiples

Ce modèle peut être étendu afin de simuler plusieurs vagues déferlantes. La force d'impulsion appliquée à une particule est simplement décrite par une superposition de plusieurs

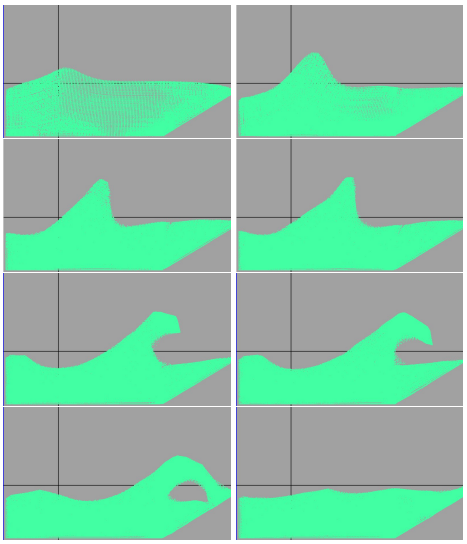


Figure 1: *Les différentes phases du déferlement d'une vague en 2D calculé avec notre méthode*

solitons :

$$F^{imp} = \sum_{i}^{N} F_i^{soliton} \tag{7}$$

où N est le nombre de solitons présents dans la scène et $F_i^{soliton}$ désigne la force d'impulsion associée à un soliton i.

La somme des forces appliquée à une particule donnée peut alors s'exprimer à l'aide de la formule suivante :

$$F = F^{press} + F^{visc} + F^{ext} + F^{imp} \tag{8}$$

Pour reproduire le déferlement d'un train de vagues, il suffit de créer plusieurs solitons et donc plusieurs forces d'impulsions, en regénérant à chaque fois un triplet aléatoire (H_i, c_i, ϕ_i) nous permettant d'obtenir un résultat non uniforme temporellement. Cette étape s'effectue si la somme des forces d'impulsions appliquées au fluide est proche de 0.

Notre méthode permet ainsi d'éviter une limitation temporelle, et autorise en outre un contrôle des caractéristiques de chaque train de vagues déferlantes. Un utilisateur peut ainsi définir la hauteur et la phase d'un train de vagues, tout en contrôlant leur nombre simulé dans la scène.

5. Simulation des sprays

Le déferlement des vagues est un phénomène s'accompagne d'une intense interaction entre l'atmosphère et la la surface de l'océan, se caractérisant par la génération de sprays. Pour décrire ce phénomène, les méthodes existantes ont recours à l'ajout de systèmes de particules le plus souvent contrôlés de manière indépendante.

Nous proposons ici une méthode de génération automatique des sprays spécifique au déferlement d'une vague, en représentant la gazéification en air d'une particule d'eau en fonction de la distance de la particule vis à vis de la surface libre et de sa quantité de mouvement.

Le calcul de la densité au repos ρ_0 des particules est remplacé par la résolution du système :

$$\frac{\partial \rho_0}{\partial t} = (|\frac{\vec{\partial u}}{\partial t}| - U_s)(\rho_{0_{air}} - \rho_{0_{eau}})$$
$$\rho_{0_{t+1}} = \rho_{0_t} + \frac{\partial \rho_0}{\partial t}\Delta t$$

où U_s désigne la quantité de mouvement suffisante pour induire un changement d'état et $\rho_{0_{eau}}$ (resp. $\rho_{0_{air}}$) la densité au repos de l'eau (resp. de l'air). Nous appliquons un schéma d'intégration explicite et nous bornons la valeur obtenue entre la densité au repos de l'eau et de l'air.

La quantité d'air présent dans une particule est alors calculée par :

$$Q_{air} = \frac{\rho_{0_i} - \rho_{eau}}{\rho_{air} - \rho_{eau}} \tag{9}$$

Si une particule passe en phase gazeuse, nous générons un sous-système de particules représentant des sprays où chaque sous-particule est considérée comme une fille de la particule mère. Aucune interaction entre les sous-particules n'est calculée afin de limiter les calculs : leur mouvement est conduit dynamiquement par le mouvement de leur mère au niveau hiérarchique supérieur. Dans le cas contraire, si une particule "gazéifiée" tend à passer vers un état liquide, nous supprimons aléatoirement des particules de sprays en fonction de la quantité d'air présent dans la particule mère.

Les sprays sont ensuite rendus sous forme de blobs dont le rayon est généré aléatoirement et dont l'opacité est calculée en fonction de la quantité d'air de la particule mère. Ainsi, plus la quantité de mouvement d'une mère proche de la surface est importante, plus le nombre de particules de sprays générées sera grand (et plus celles-ci apparaîtront opaques). Dans le cas contraire, plus une particule mère tend à passer vers une phase liquide, plus ses sprays apparaîtront transparents (voir Fig. 2).

6. Multirésolution lagrangienne

La multirésolution lagrangienne apporte une solution au problème du contrôle du coût en mémoire et en temps de calcul en permettant une représentation hiérarchique du système global à travers deux opérations élémentaires appelées "fusion" et "division" qui consiste respectivement à rattacher ou détacher des particules au sein d'une même particule.

Cependant, les approches existantes nécessitent des calculs coûteux de champs de distance vis à vis de la surface libre ainsi qu'un redistancement des particules. Nous proposons donc une simplification spécifique au

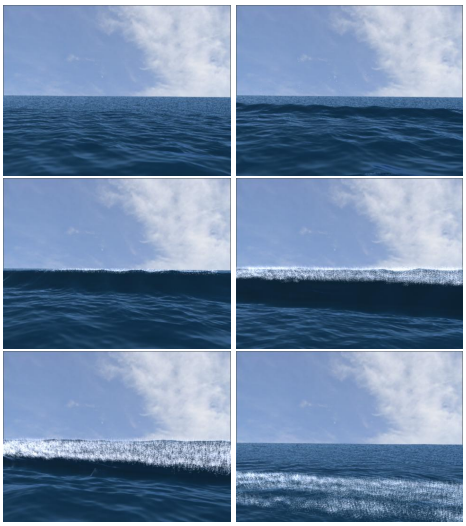


Figure 2: *Génération des sprays pendant le processus de déferlement*

déferlement des vagues afin de limiter les calculs.

Comme mentionné dans [APKG07], nous sommes confrontés au problème de la conservation de la masse totale pendant les opérations de fusion et de division. Le principe de notre méthode est un transfert de masses à chaque pas de temps, consistant à définir l'opération de fusion comme l'augmentation progressive de la masse d'une particule tout en réduisant celles de particules identifiées comme ses filles ; l'opération de division est définie par le processus inverse. La masse perdue par une particule est donc toujours gagnée par une autre, ce qui garantit ainsi la conservation de la masse totale. Ce processus est illustré sur la figure 3. Les rayons d'interaction des particules mère et filles sont ensuite recalculés par la formule 6.

Nous utilisons la méthode *shooting / gathering* de Desbrun et Cani [DC99] utilisée également dans [APKG07] pour calculer les forces de pression et de viscosité afin de respecter la symétrie des forces entre deux particules de tailles différentes.

6.1. Fusion

Notre processus de fusion s'effectue en deux étapes.

En premier lieu, les particules du système pouvant être affectées par cette opération sont identifiées en tenant compte de différents critères : la profondeur de la particule vis à vis de l'interface air/eau, la force d'impulsion de déferlement à laquelle est soumise la particule et la distance de la particule par rapport à l'observateur. En effet, nous supposons que plus une particule est proche de l'interface air/eau, moins nous pouvons approximer le calcul des forces

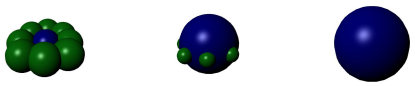


Figure 3: *Transfert de masse pendant une opération de fusion entre la particule mère (au centre) et ses filles sur des images successives*

en raison notamment des tensions de surface exercées à l'interface. De manière similaire, plus la force d'impulsion de déferlement à laquelle est soumise une particule est importante, moins nous pouvons approximer ses échanges. Enfin, comme chaque particule possède un impact visuel sur le rendu final de la simulation, plus une particule se trouve éloignée du point de vue, plus nous pouvons approximer son comportement dynamique.

Notons H_i la profondeur d'une particule i vis à vis de la surface libre, et H_{min} (resp H_{max}) la profondeur minimale (resp maximale) ; ces paramètres globaux sont fixés au départ de la simulation. Notre critère de profondeur est alors $C_{H_i} = |\frac{H_i - H_{min}}{H_{max} - H_{min}}|$, les critères de distance à la caméra C_{D_i} et de force d'impulsion $C_{F_i^{imp}}$ étant définis de façon similaire (resp. à partir de la distance D_i et de la force d'impulsion verticale $F^{imp}i$ à laquelle est soumise la particule). Une différence notable est que la force d'impulsion verticale maximale n'est pas un seuil défini arbitrairement mais est définie comme le maximum de la somme des forces d'impulsions pouvant s'appliquer sur une particule.

Le critère de choix est simplement défini par $C_i = C_{H_i} + C_{D_i} + C_{F_i^{imp}}$; on considérera qu'une particule peut attacher d'autres particules si $C_i > 1$. Si c'est le cas, les particules qui lui sont attachées sont celles se trouvant à une distance inférieure à h de la particule identifiée et qui respectent elles aussi le critère de choix.

Après cette identification, la seconde étape du processus consiste à transférer une partie de la masse des particules filles vers la particule mère. A chaque pas de temps, chaque particule fille perd une faible quantité de masse rétribuée à sa mère. Pour chaque particule fille et mère, nous recalculons ensuite un nouveau rayon d'interaction ; ce processus s'arrête dès que la particule mère a atteint sa masse maximale, calculée comme la somme de sa masse initiale et des masses initiales des particules filles pondérées par leur nombre.Une fois atteint ce critère, nous désactivons les particules filles au sein du système : elles n'interviendront notamment plus dans les calculs de voisinage représentant l'essentiel du coût de la simulation.

6.2. Division

Le processus de division est l'inverse du processus de fusion :une particule ne pourra plus représenter dynamique-

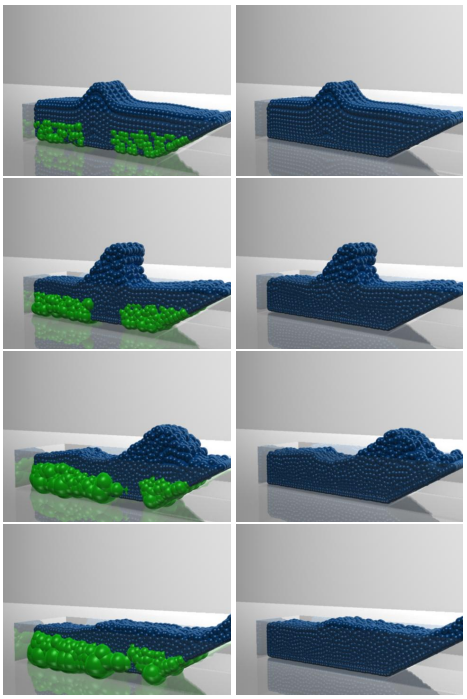


Figure 4: *Comparaison de la simulation obtenue avec et sans multirésolution avec 4000 particules initiales et un soliton de paramètre* $(0.35, 0, 0)$

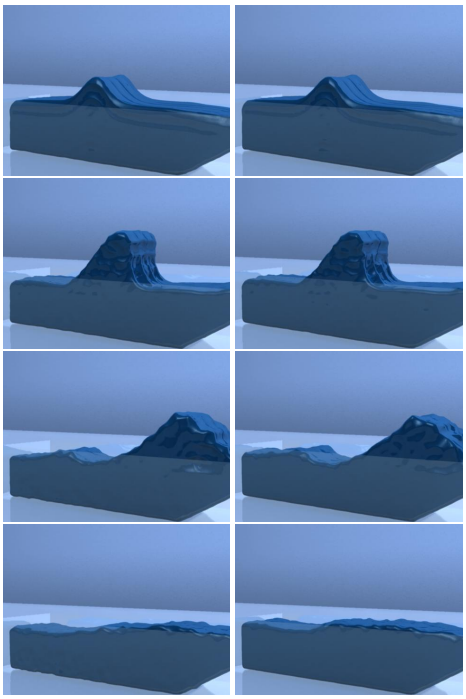


Figure 5: *Comparaison de la surface obtenue à partir de la simulation de la figure précédente*

ment ses filles si elle a subi une opération de fusion et si son critère de choix $C_i <= 1$.

Si c'est le cas nous transférons à chaque pas de temps une faible quantité de masse de la particule vers chacune de ses filles, et nous arrêtons le processus lorsque la particule mère est revenue à sa masse initiale.

Notre méthode permet ainsi d'obtenir un nombre réduit de particules actives nécessaires à une simulation de vagues déferlantes : les zones d'importance sont échantillonnées par des particules de taille minimale, alors que dans les zones moins significatives cette taille croît linéairement pendant la simulation, comme le montrent les images prises à différentes étapes de la simulation montrant les tailles adaptives des particules (Figs. 4 et 5).

Le transfert linéaire de masse entre particules filles et mères nous garantit le respect de la conservation de la masse durant le processus et ainsi la stabilité du système global. De plus, l'utilisateur peut contrôler à travers le nombre initial de particules la mémoire maximale nécessaire pour la simulation, puisqu'aucune nouvelle particule n'est créée par notre méthode.

7. Résultats

Pour observer les gains obtenus par notre approche, plusieurs simulations ont été menées sur un processeur AMD64 à 2,4 Ghz et 1 Go de mémoire. Le gain apporté par la multirésolution, en nombre de particules désactivées, croît en fonction du nombre de particules présent dans la simulation ; de façon similaire, les temps de calculs sont accélérés de façon linéaire (Fig. 6). Par exemple, pour une simulation avec 60K particules initiales, nous obtenons un gain de 40% en termes de particules désactivées, et ainsi une simulation $1,5$ fois plus rapide (environ 62 sec. par image en moyenne avec la multirésolution contre 108 en simulation standard).

Comme le montre la figure 7, les évolutions de la moyenne des densités et des forces au cours d'une animation de 600 images avec multirésolution est comparable à celles d'une simulation sans multirésolution, ce qui confirme le résultat visuel des figures 4 et 5.

Les oscillations observées dans l'évolution de la moyenne des densités sont principalement dues aux effets de compressibilité, et représentent le principal point que nous souhaitons améliorer.

La figure 8 montre des images extraites d'une séquence vidéo de 600 images, générée à l'aide de 100K particules initiales avec 2 solitons présents dans la scène de paramètres respectifs $H = (0.35, 0.3)$, $c = (0.2, 0.1)$ et $\phi = (1.4, 0.0)$. Afin

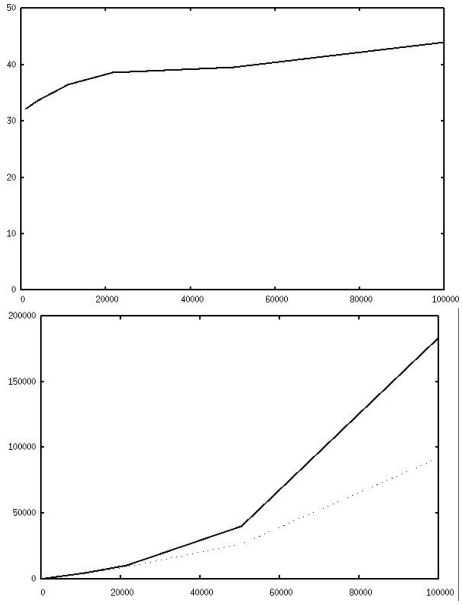


Figure 6: *En haut, gain amené par la multirésolution en pourcentage de particules désactivées sur une simulation complète de 600 images en fonction du nombre de particules. En bas, comparaison de l'évolution des temps de calcul entre une simulation standard (trait plein) et multirésolution (trait pointillé)*

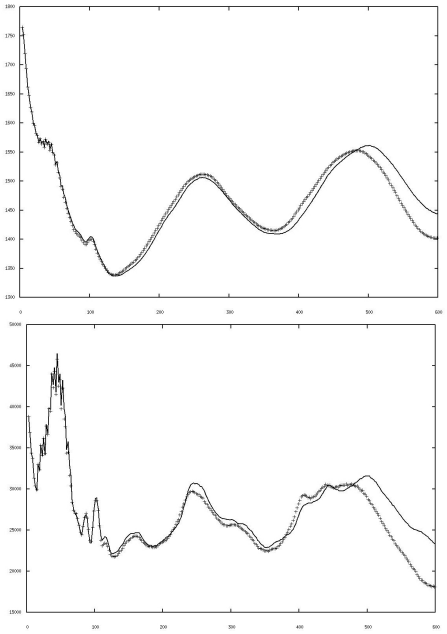


Figure 7: *Comparaison de l'évolution de la moyenne par image des densités (en haut) et de la moyenne des forces (en bas) sur l'ensemble des particules actives de la simulation. La simulation standard est en trait plein, celle multirésolution en trait pointillé*

d'extraire l'iso-surface définie par les particules, nous utilisons l'algorithme du logiciel commercial Glu3D (`http://3daliens.com/`), puis le lancer de rayons MentalRay (`http://www.autodesk.fr/`) pour le rendu de la surface et des sprays générés par le déferlement.

8. Conclusion

Nous avons présenté une nouvelle méthode permettant de simuler des vagues déferlantes. Notre approche, basée sur la superposition d'impulsions finies représentées sous forme de forces extérieures au sein du solveur, permet de capturer ce phénomène sans les contraintes temporelles des méthodes existantes. Nous proposons également d'enrichir le rendu par une méthode de génération automatique des sprays qui permet d'obtenir des résultats plus réalistes sans accroître la complexité. Enfin, pour permettre une simulation à grande échelle, nous avons introduit un schéma multirésolution permettant de contrôler le coût mémoire et le temps de calcul.

Ces travaux se placent dans une perspective plus générale de rendu réaliste de scènes océaniques. C'est pourquoi notre approche devra intégrer la génération d'autres phénomènes liés aux turbulences (principalement l'écume); notre schéma multirésolution pourrait lui aussi être adaptée au cas plus généraux liés à la présence de rochers, bras de mer, etc.

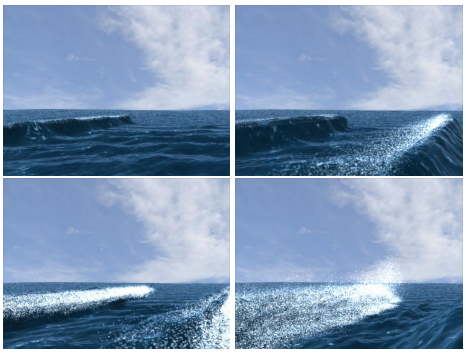


Figure 8: *Simulation complète avec 100K particules*

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. In *ACM Transactions on Graphics (SIGGRAPH '07 papers)* (New York, NY, USA, 2007), vol. 26, ACM Press, pp. 48–48*.

[CBP05] CLAVET S., BEAUDOUIN P., POULIN P.: Particle-based viscoelastic fluid simulation. pp. 219–228.

[CGG01] CIEUTAT J. M., GONZATO J. C., GUITTON P.: A new efficient wave model for maritime training simulator. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics* (Washington, DC, USA, 2001), IEEE Computer Society, p. 202.

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. *In 6th Eurographics Workshop on Computer Animation and Simulation '96* (1996), 62–76.

[DC99] DESBRUN M., CANI M.-P.: *Space-Time Adaptive Simulation of Highly Deformable Substances*. Tech. Rep. 3829, INRIA, BP 105 - 78153 Le Chesnay Cedex - France, December 1999.

[DP86] DARWYN R., PEACHEY: Modeling waves and surf. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 65–74.

[EMF02] ENRIGHT D., MASCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Transaction on Computer Graphics 21 3* (2002), 736–744.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. *In Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques, ACM Press.* (2001), 23–30.

[FR86] FOURNIER A., REEVES W. T.: A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 75–84.

[IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph. 25*, 3 (2006), 805–811.

[JBS03] JESCHKE S., BIRKHOLZ H., SHUMANN H.: A procedural model for interactive animation of breaking ocean waves. *Computer Graphics* (2003).

[KCC*06] KIM J., CHA D., CHANG B., KOO B., IHM I.: Practical animation of turbulent splashing water. *Eurographics / ACM SIGGRAPH Symposium on Computer Animation* (2006).

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. *Computer Graphics* (1990), 49–57.

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *Computer Graphics* (2004).

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (2006), 812–819.

[MCG03] MULLER M., CHARYPAR P., GROSS M.: Particle-based fluid simulation for interactive applications. *In Proceedings of 2003 ACM SIGGRAPH Symposium on Computer Animation* (2003), 154–159.

[MMS04] MILHAEF V., METAXAS D., SUSSMAN M.: Animation and comtrol of breaking waves. *Proceedings of the 2004 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2004), 315–324.

[Mon92] MONAGHAN J.: Smoothed particle hydrodynamics. *Annu. Rev. Astron. Physics 30 543* (1992).

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. *In Proceedings on Symposium on Computer Animation* (2005), 237–244.

[OH95] O'BRIEN J., HODGINS J.: Dynamic simulation of splashing fluids. *Computer Animation 95* (1995), 188–205.

[PTB*03] PREMOZE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. In *Proceedings of Eurographics 2003* (2003), pp. 401–410.

[QYC*06] QIANG W., YAO Z., CHUN C., FUJIMOTO T., NORISHIGE C.: Efficient rendering of breaking waves using mps method. *Journal of Zhejiang University* (2006).

[RO98] RADOVITZKY R., ORTIZ M.: lagrangian finite element analysis of newtonian fluid flows. *Int J. Numer. Meth. Engng 43* (1998), 607–619.

[TDG00] THON S., DISCHLER J., GHAZANFARPOUR D.: Ocean waves synthesis using a spectrum-based turbulence function. In *Computer Graphics International* (2000).

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. In *Eurographics* (2003).

[TMFSG07] THUEREY N., MUELLER-FISCHER M., SCHIRM S., GROSS M.: Real-time breaking waves for shallow water simulations. In *Pacific Graphics* (2007).

[TRS06] THUEREY N., RÜDE U., STAMMINGER M.: Animation of open water phenomena with coupled shallow water and free surface simulations. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2006).

Figure 1: Simulation complète avec 100K particules

Rendu interactif de nuages réalistes

Antoine Bouthors[1] Fabrice Neyret[1] Nelson Max[2] Eric Bruneton[1] Cyril Crassin[1]

[1]EVASION - LJK / Grenoble Universités - INRIA [2]LLNL - UC Davis

Abstract

Nous proposons un algorithme simulant interactivement la dispersion anisotrope multiple de la lumière (multiple anisotropic scattering) *dans un milieu homogène. Contrairement aux méthodes temps-réel précédentes, nous reproduisons tous les types de chemins lumineux à travers le milieu et nous préservons leur caractère anisotrope.*

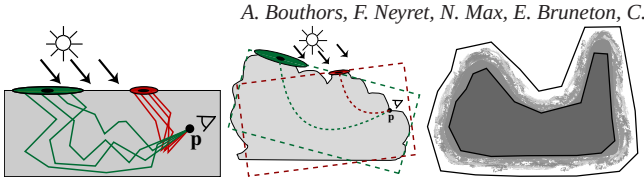
Notre approche consiste à estimer le transport d'énergie depuis la surface éclairée du milieu vers les pixels à rendre, en considérant séparément chaque ordre de dispersion. Nous représentons la distribution des chemins lumineux parvenant à un pixel donné du volume par la moyenne et l'écart-type de leur point d'entrée sur la surface illuminée, que nous appelons "aire collectrice". Au moment du rendu, nous déterminons sur la surface illuminée l'aire collectrice pour chaque pixel et pour différents groupes d'ordre de dispersion, et en déduisons le transport lumineux associé. La recherche rapide de l'aire collectrice et le calcul du transport lumineux est rendue possible grâce à une étude préliminaire de la dispersion multiple dans des formes simplifiées et ne nécessite pas de parcourir le volume.

Le rendu est réalisé efficacement dans un shader *sur le processeur graphique* (GPU)*, en utilisant un maillage de la surface du nuage enrichi par une Hypertexture ajoutant des détails à la forme. Nous présentons notre modèle avec le rendu interactif de cumulus animés et détaillés à 2-10 images par secondes.*

We propose an algorithm for the interactive realistic simulation of multiple anisotropic scattering of light in participating media. Contrary to previous real-time methods we account for all kinds of light paths through the medium and we preserve their anisotropic behavior.

Our approach consists in estimating the energy transport from the illuminated surface to the rendered pixel of the medium for each separate order of multiple scattering. We represent the distribution of light paths reaching a given viewed cloud pixel with the mean and standard deviation of their entry point on the lit surface, which we call the "collector area". At rendering time for each pixel we determine the collector area on the lit cloud surface for different sets of scattering orders, then we infer the associated light transport. The fast computation of the collector area and light transport is made possible thanks to a preliminary analysis of multiple scattering in plane-parallel slabs and does not require slicing or marching through the volume.

Rendering is done efficiently in a shader on the GPU, relying on a surface mesh augmented with a Hypertexture to enrich the shape. We demonstrate our model with the real-time rendering of detailed animated cumulus and cloudy sky at 2-10 fps.



(a) (b) (c)

Figure 1: *Vue d'ensemble de nos contributions.* (a) *Dans une analyse préliminaire, nous caractérisons le transport de la lumière dans une dalle via une aire collectrice représentant la zone d'entrée de la lumière pour chaque ordre de dispersion.* (b) *Nous utilisons cette caractérisation pour trouver les zones collectrices sur une forme de nuage quelconque et calculer le transport de la lumière.* (c) *Nos nuages sont représentés par une Hypertexture, avec des détails procéduraux sur les bords et un noyau homogène.*

1. Introduction

Le rendu réaliste de nuages est un problème difficile. La dispersion anisotrope de la lumière doit être simulée dans un volume, le manque d'absorption rend la convergence très lente, et les nuages détaillés requièrent des représentations volumiques à haute résolution. Dans l'optique du rendu temps-réel, ceci est encore plus difficile: les méthodes de radiosité volumique ou de Monte-Carlo ne peuvent pas converger en temps-réel, et le rendu volumique ne peut pas être effectué avec suffisamment de résolution pour des nuages détaillés tels que les cumulus. Les précalculs empêchent l'animation des nuages ou de la source de lumière, et la plupart des modèles temps-réel ne reproduisent pas les effets dépendant du point de vue ou des aspects visuellement importants tels que la rétro-dispersion (*backscattering*).

Pour résoudre ces problèmes, notre approche est de représenter les nuages par des volumes délimités par des surfaces et optimise le calcul du transport lumineux depuis la surface illuminée du nuage vers les pixels à rendre. Pour cela, nous étudions et caractérisons le transport de la lumière pour chaque ordre de dispersion.

Nos contributions sont:

- un nouveau modèle caractérisant le transport de la lumière (*i.e.*, la quantité d'énergie lumineuse transmise) entre la surface d'une dalle homogène de nuage (*i.e.*, un volume de nuage limité par deux plans parallèles) et un point **p** quelconque dans la dalle, pour chaque ordre de dispersion (voir figure 1(a));
- un nouveau modèle caractérisant la distribution, sur la surface d'une dalle, des points d'entrée des chemins lumineux arrivant en **p**, pour chaque ordre de dispersion. Nous appelons cette zone d'entrée l'*aire collectrice* ou collecteur (voir figure 1(a));
- un algorithme itératif déterminant cette aire collectrice sur un nuage de forme quelconque pour un point **p** quelconque, et calculant le transport lumineux associé (voir figure 1(b));
- une représentation efficace de la forme détaillée d'un nuage utilisant une Hypertexture [PH89] sur un maillage surfacique (voir figure 1(c));
- une implémentation sur GPU de ces contributions, résultant en un rendu détaillé de nuages animés en temps interactif prenant en compte la dispersion anisotrope multiple de la lumière à tous les ordres.

Contrairement à la plupart des méthodes temps-réel, notre modèle de dispersion multiple

- prend en compte les ordres élevés de dispersionj responsables des effets de diffusion et de rétro-dispersion, ainsi que les faibles ordres de dispersion responsables de la gloire, des silhouettes lumineuses et de l'apparence des parties fines;
- utilise la fonction de phase de Mie fortement anisotrope, basée sur la physique de la dispersion de la lumière par les petites particules (et non pas une fonction de Rayleigh, une Gaussienne, ou une fonction isotrope);
- ne nécessite pas de parcourir le volume du nuage, mais seulement sa surface;
- ne se base pas sur des précalculs dépendants de la forme du nuage.

Représenter la forme du nuage par une Hypertexture sous un maillage de surface nous permet de rendre efficacement des nuages dont les bords sont hétérogènes aussi bien que nets, contrairement aux méthodes basées sur des volumes tranchés. Nous pouvons ainsi rendre rapidement des nuages nets, cotonneux ou vaporeux.

2. Physique des nuages

2.1. Densité et taille des gouttelettes de nuages

Les nuages convectifs réels ne sont pas flous sur leur bords, mais nets ou effilochés (voir figures 5(c) et 5(d)). Les parties où la convection est faible peuvent avoir une couche de filaments plus large. Ces hétérogénéités dans le contenu en eau d'un nuage (*i.e.*, sa densité) constituent un aspect visuel fort des nuages. A l'intérieur d'un nuage, cette densité peut aussi varier – en particulier à l'approche des conditions de pluie – à cause de la coalescence des gouttelettes.

La taille de ces gouttelettes est caractérisée en un point donné par une distribution de taille de gouttelettes (*droplet size distribution* – DSD), qui est généralement modélisée dans la littérature par une distribution log-normale ou Gamma modifiée [Lev58]. Les propriétés optiques d'un nuage dépendent fortement de la taille des gouttelettes. Prendre la DSD en compte change radicalement la fonction de phase résultante, et donc l'apparence visuelle.

Cependant, ces données pour un nuage ne sont pas souvent disponibles et la physique qui les régit (*e.g.*, le mécanisme de coalescence ou l'évolution de la DSD) n'est pas totalement comprise. Étant donné que les applications en informatique graphique requièrent en priorité une plausibilité visuelle, les approximations de ces valeurs sont classiques. À titre d'exemple, les variations de densités sont visuellement plus importantes sur les bords des nuages – où elles sont directement visibles – qu'à l'intérieur – où elles n'influence l'apparence qu'indirectement, via les grands ordres de dispersion. Habituellement, la DSD est considérée comme constante pour tout un nuage (si tant est qu'une DSD soit utilisée).

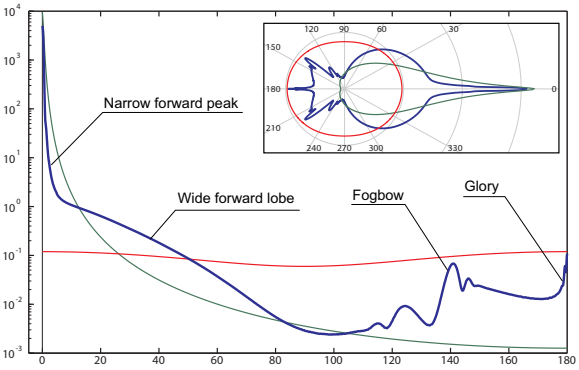


Figure 2: *Graphe logarithmique (encart: graphe logarithmique en coordonnées polaires) de fonctions de phases couramment utilisées.* Rouge: *Rayleigh.* Vert: *Henyey-Greenstein avec* $g = .99$. Bleu: *Mie.*

2.2. Fonction de phase

La fonction de phase d'une gouttelette d'eau est déterminée par la théorie de Mie et ne peut pas être approximée précisément par une Gaussienne, une fonction de Henyey-Greenstein, ou d'autres modèles. Comme on peut le voir figure 2, la fonction de Mie combine un fort pic en avant (51% de l'énergie), un large lobe en avant (48%), et un lobe arrière complexe comprenant des pics. Ces trois caractéristiques apportent chacune des effets très spécifiques et visibles à l'échelle du nuage. Une absence de pics arrières signifie que l'on n'aura ni gloire, ni arcs blancs (sortes d'arc-en-ciel créés par les nuages). Ignorer le pic avant implique une forte sous-estimation de la transmittance globale et de l'anisotropie. Les fonctions Gaussienne et Henyey-Greenstein contiennent bien un lobe en avant et facilitent les calculs, mais sont loin de donner des résultats visuels précis. La fonction de dispersion de Rayleigh est encore moins appropriée puisqu'elle est symétrique (50% de la dispersion est en arrière) et correspond à la dispersion par les molécules de l'air (qui donnent la couleur bleue au ciel), et non pas par les gouttelettes d'eau.

Dans le spectre du visible, l'albédo d'une gouttelette peut être considéré comme valant 1 puisque'il n'y a pas d'absorption (toute la lumière est dispersée). À noter que certains phénomènes atmosphériques communément attribués aux nuages sont en fait causés par d'autres éléments (*e.g.*, les arc-en-ciel sont produits par la pluie, les parhélies par les cristaux de glace en suspension dans l'atmosphère).

2.3. Anisotropie

La dispersion multiple est forte dans les nuages et montre souvent un caractère anisotrope. Les nuages couvrent des centaines, voire des milliers de mètres et le libre parcours moyen d'un rayon de lumière dans un nuage est d'environ 20 m. En conséquence, la plupart des rayons seront dispersés de multiples fois avant de sortir du nuage. A cause de la nature hautement anisotrope de la fonction de phase de Mie

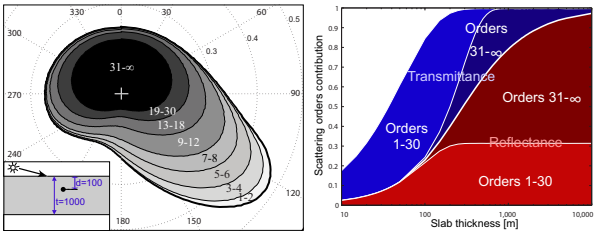


Figure 3: *Quelques résultats de notre analyse du transport de la lumière.* **Gauche :** *BSDF pour un point de vue à une profondeur de* $d = 100m$ *à l'intérieur d'une dalle de nuage d'épaisseur* $t = 1000m$*, avec un angle incident d'illumination de* $\phi_L = 75°$*. Les aires représentent la contribution de différents ordres de dispersion. Même les ordres élevés (jusqu'à 30 événements de dispersion) montrent un comportement anisotrope, tandis que les ordres* > 30 *montrent un comportement isotrope.* **Droite :** *Contribution des différents ordres de dispersion pour la réflectance (en rouge) et la transmittance (en bleu) d'une dalle d'épaisseur variable. Les ordres isotropes (31-*∞*) commencent à apparaître aux épaisseurs* $> 100m$*. Les ordres anisotropes (1-20) jouent un rôle dans la transmittance jusqu'à des épaisseurs de* $1000m$*, et contribuent à la réflectance à hauteur de* $30\% - 100\%$*.*

(99% de la lumière est dispersée en avant), même la dispersion multiple peut être anisotrope. D'après notre analyse du transport de la lumière dans une dalle, nous estimons que le comportement de la lumière est isotrope seulement après au moins 30 événements de dispersion. Par isotropie, nous entendons que la dispersion multiple se comporte comme si la fonction de phase du milieu était isotrope, et non que la BSDF (*Bidirectional Scattering Distribution Function*, fonction décrivant la distribution de la lumière selon l'angle de vue en un point donné) résultante est isotrope. En conséquence, l'illumination d'un nuage n'est pas dominée par les faibles ordres de dispersion, mais les ordres élevés ne se comportent pas tous de façon isotrope (voir figure 3).

3. État de l'art

Simuler la dispersion anisotrope multiple par des méthodes classiques de Monte-Carlo ou de radiosité volumique n'est pas possible en temps-réel. Les optimisations actuelles se basent sur des simplifications variées, que nous présentons dans cette section.

3.1. Fonction de phase

Étant donné que la fonction de Mie est complexe et coûteuse à calculer, elle est souvent approximée en synthèse d'images par d'autres fonctions comme la fonction de Henyey-Greenstein [Max94], une Gaussienne [PAT*04] ou même la fonction de Rayleigh [HL01] (voire commentaires en section 2.2). [REK*04, BNL06] précalculent la fonction de phase de Mie pour une DSD donnée, ce qui reproduit les aspect de la réalité (gloire, arc blancs, etc.). Nous utilisons la même approche pour notre fonction de phase.

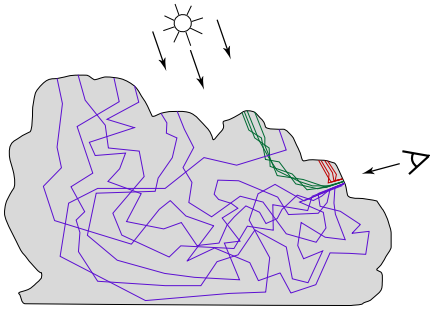


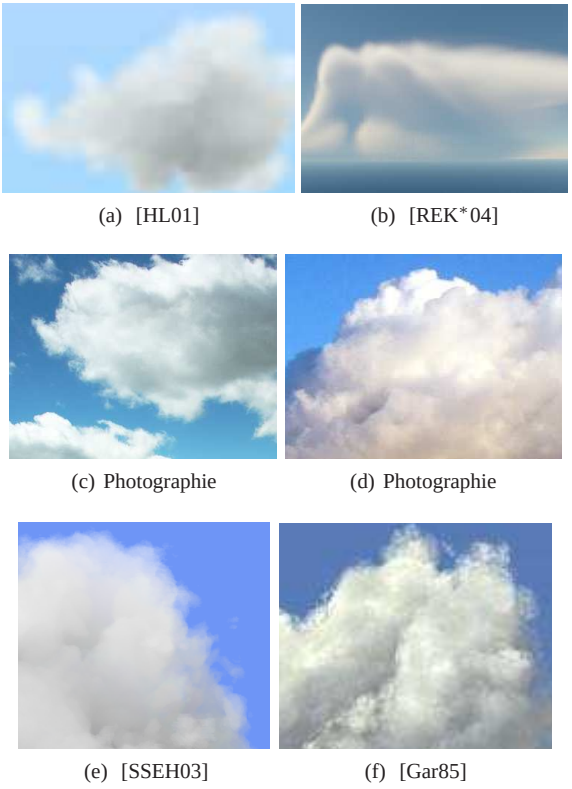
Figure 4: *Différent types de chemins lumineux dans les milieux participants, apportés par différents ordres de dispersions. Les faibles ordres de dispersion apportent des chemins courts et très courbés (en rouge). Les ordres plus élevés apportent des chemins longs et peu courbés (en vert). Les ordres les plus élevés apportent des chemins complexes, très étalés et diffusifs (en bleu). L'approche des* chemins les plus probables *[PAS03] reproduit les chemins verts, mais sous-estime les chemins rouges et bleus.*

3.2. Transport de la lumière

[Kv84] et [Bli82] considèrent soit un albédo faible (la lumière est peu diffusée), soit une densité faible. Dans ces cas, seul le *single scattering* (on ne prend en compte que les rayons ayant été dispersés une seule fois) est considéré. Ceci supprime tous les effets dus à la dispersion multiple. L'hypothèse que le transport de la lumière est principalement vers l'avant amène des algorithmes temps-réel en une seule passe telle que l'accumulation de tranches du volume [DKY*00, HL01, REK*04] mais empêche certains effets de la dispersion multiple tels que la rétro-dispersion. Calculer la dispersion multiple [NND96] seulement pour les ordres les plus faibles a les mêmes conséquence. L'approximation de diffusion [Sta95, JMLH01] permet des calculs efficaces mais néglige l'anisotropie dans la dispersion multiple.

[PAS03] ont introduit l'idée de *chemins les plus probables* (*Most Probable Paths* – MPP) dans les milieux participants. L'idée principale est que la majorité des photons arrivant à un point dans une direction ont grossièrement suivi le même chemin. En conséquence, intégrer le transport de la lumière uniquement le long de ce chemin est suffisant pour prendre en compte la majeure partie du transport. De plus, [PAT*04] accélèrent cette technique et tiennent compte de l'étalement spatial de la lumière autour de ce chemin moyen à travers une formulation analytique. Cette approche a été portée au temps-réel [HAP05] en utilisant le matériel graphique pour trancher le volume dans une manière similaire à [HL01, REK*04]. Ces méthodes basées sur deux passes de tranchage (accumuler le flux depuis la source de lumière dans les tranches, puis des tranches vers l'œil) limitent la variété de chemins pris en compte.

Comme indiqué dans [PAS03, HAP05], la principale restriction de l'approche MPP est que les chemins qu'elle calcule sont principalement d'ordre élevé et peu courbés. En conséquence, les chemins d'ordre faible, ainsi que les chemins diffusifs, sont sous-estimés. Ces chemins contribuent à la majorité de la luminance dans les parties fines et dans la rétro-dispersion, et ne devraient donc pas être négligés (voir figure 4).



(a) [HL01] (b) [REK*04]

(c) Photographie (d) Photographie

(e) [SSEH03] (f) [Gar85]

Figure 5: En haut: *cumulus virtuels temps-réels* **(a)** *et interactifs* **(b)** *utilisant des* billboards *ou des tranches.* Au milieu: *Cumulus réels effilochés* **(c)** *et nets* **(d)**. En bas: *Nuages virtuels utilisant un bruit 3D* **(e)** *et des surfaces avec détails procéduraux* **(f)**. *Ajouter des détails procéduraux donne une apparence moins floue et plus contrastée, augmentant le réalisme.*

Nous abordons ces restrictions en traitant les chemins lumineux de tous les ordres. Nous proposons également une nouvelle approche du calcul du transport lumineux, plus rapide, qui ne requiert pas de trancher ou parcourir le volume. Notre approche est également inspirée par les études sur le transfert radiatif dans des formes simples telles que les dalles [Cha60, KE86, Mob89, MMKW97]. Nous utilisons des représentations adaptées au matériel graphique telle que les *depth maps* (textures de profondeur) [DS03] pour implémenter notre algorithme de rendu sur GPU.

3.3. Représentation des champs de densités des nuages

Étant donné qu'il est difficile de mesurer ou simuler les données de densité de nuages, les premiers travaux ont

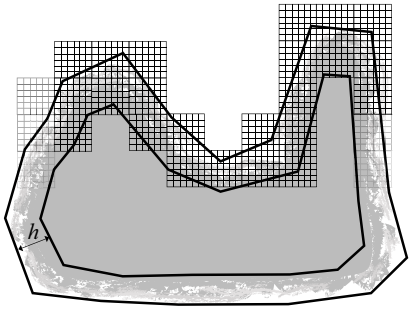


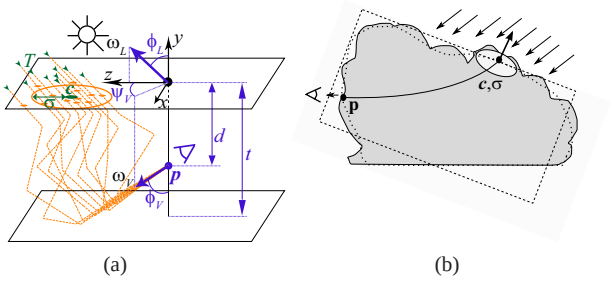
Figure 6: *Notre représentation de nuages. Un maillage est utilisé pour décrire la bordure extérieure à basse résolution. Une Hypertexture procédurale volumique ajoute des détails sous la bordure jusqu'à une certaine profondeur h à l'intérieur du nuage. Le cœur est considéré comme homogène.*

utilisé des modèles procéduraux [Gar85, DKY*00]. Les approches récentes utilisent des techniques de simulation de fluides [HL01] ou des données atmosphériques [TB02, REK*04]. Cependant, la simulation de fluides en 3D ne peut être faite qu'a une résolution grossière dans le cas d'applications temps-réel, et les données atmosphériques sont à basse résolution. En conséquence, il est nécessaire d'ajouter des détails de haute fréquence afin d'éviter une apparence floue (voir figure 5). [Ebe97] combine un bruit de Perlin [Per85] solide et des surfaces implicites. [SSEH03] advectent une texture de bruit [Ney03], mais ne tiennent pas compte du bruit dans le calcul d'illumination, ce qui donne au nuages une apparence uniforme (voir figure 5(e)). Ces idées nous ont inspiré pour combiner deux représentations – des maillages et des textures 3D – pour modéliser les nuages à deux échelles différentes.

Des primitives de rendu variées ont été utilisées pour rendre des nuages. Étant donné que les nuages sont une distribution spatiale de gouttelettes, des grilles volumiques ont souvent été utilisées pour les décrire, et les techniques de rendu volumique pour les rendre. Les méthodes utilisant des *billboards* ou des tranches de volume [HL01, PAT*04, REK*04, DKY*00] résultent en beaucoup d'*overdraw*[1], ce qui a un coût de rendu important. Si les tranches texturées sont une façon efficace de rendre des phénomènes gazeux, elles ne sont pas la meilleure options pour les nuages denses, où la plupart des pixels des tranches lointaines sont cachés par les plus proches, et donc inutiles à rendre. De plus, la mémoire disponible pour les textures 3D limite la résolution de tels modèles, ce qui résulte en des silhouettes floues et un manque de détails (voir figures 5(a), 5(b)).

Étant donné que les cumulus sont denses et ont souvent une interface nette, ils ont également été représentés par des volumes limités par des surfaces, tels que des ensembles d'ellipsoïdes [Gar85, ES00] ou des maillages [TB02].

[1] *i.e.*, un pixel donné est *rasterisé* beaucoup de fois par différentes primitives rendues.



(a) (b)

Figure 7: **(a)** *Schéma et notation pour le transport lumineux canonique T entre une aire collectrice $(\mathbf{c},\sigma)$ et un point* **p** *dans une dalle, dépendant des paramètres d'entrées $(\phi_V,\psi_V,\phi_L,d,t)$ pour chaque ordre de dispersion. À noter que ψ_L est toujours égal à 0 (le repère de référence est aligné avec la direction de la lumière).* **(b)** *Transport de la lumière dans un nuage entre sa surface éclairée et un point* **p**. *Pour chaque groupe d'ordres de dispersion, la lumière arrivant en* **p** *est considérée comme arrivant d'une aire collectrice. Nous caractérisons ce transport en associant une dalle à cette aire collectrice, puis en utilisant la fonction de transport canonique pour cette dalle.*

Pour éviter l'apparence "dure" des surface polygonales, on utilise un *shader* procédural simulant la silhouette détaillée et donnant une impression de volume (voir figure 5(f)). Le transport de la lumière n'est pas simulé. [BNL06] simulent le transport de la lumière à l'intérieur d'un maillage, mais ne considèrent aucun enrichissement sur les silhouettes, qui apparaissent ainsi polygonales et opaques.

Nous tirons parti des deux représentations (volumes et surfaces) en représentant les bordures des nuages à grande échelle par un maillage, et les variations de densité à haute fréquence sur les bords des nuages par une Hypertexture [PH89]. Le reste de l'intérieur du nuage est considéré homogène, ainsi seulement une mince couche de voxels est nécessaire (voir Figure 6).

4. Vue d'ensemble de notre méthode

Notre approche de rendu est basée sur une analyse du transport de la lumière pour chaque ordre de dispersion. Observant que les chemins d'ordres différents ont des comportements d'anisotropie et d'étalement différents, nous les traitons séparément:

- L'ordre 1 (*single scattering*), qui est le plus anisotrope et dépend des détails les plus fins, est calculé en utilisant une forme analytique de l'équation de dispersion (voir section 6.2);
- Les ordres 2-∞ (dispersion multiple) sont calculés en 8 groupes différents (2, 3-4, 5-6, 7-8, 9-12, 13-18, 19-30, 31-∞) en utilisant notre algorithme basé sur l'aire collectrice (voir section 6.1);
- L'opacité est calculée en intégrant la fonction d'extinction à travers le volume du nuage (voir section 6.3).

Notre approche basée sur l'aire collectrice suit et étend l'idée des *chemins les plus probables* (MPP) [PAS03]. Nous

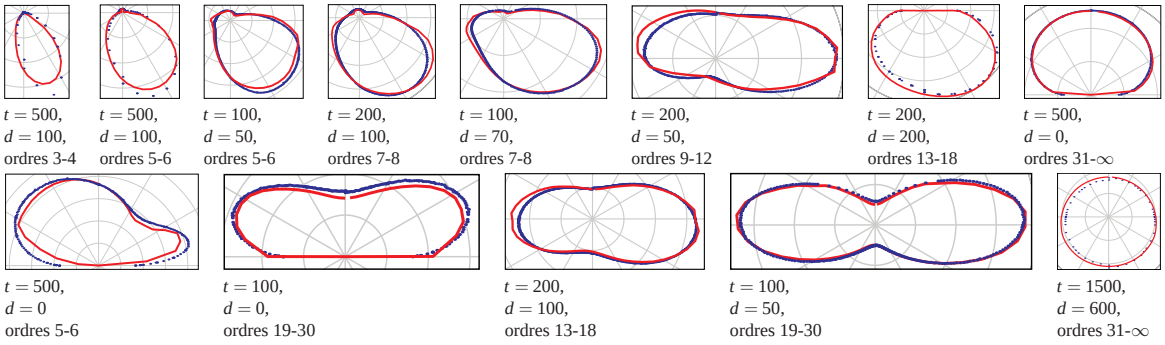


Figure 8: *Quelques BSDF résultant de notre étude du transport de la lumière. T est tracée en fonction de ϕ_V (ψ_V est fixé à $0°$) pour différentes valeurs d'entrées ($\phi_L = 15°$).* En bleu : *simulations de Monte-Carlo.* En rouge : *notre ajustement.*

considérons un chemin le plus probable et son étalement associé par groupe d'ordres de dispersion. Plus spécifiquement, nous considérons une *aire collectrice*, qui est la portion de surface à travers laquelle entre 95% de la lumière parvenant au pixel rendu dans la direction de vue (voir figure 7(b)). Cette aire collectrice est définie par son centre $\mathbf{c}$ sur la surface illuminée du nuage et par sa taille σ. Pour un pixel donné, et pour chaque groupe d'ordres de dispersion, nous cherchons ce collecteur et calculons le transport lumineux correspondant.

Pour trouver ce collecteur sur la surface illuminée, nous utilisons un algorithme qui, itérativement, associe le collecteur le plus probable à la surface du nuage. Cet algorithme est décrit en section 6.1.

Par définition, le rôle de la surface du nuage en dehors du collecteur est négligeable. Nous approchons donc localement la forme du nuage par une dalle alignée sur le collecteur afin de simplifier le calcul du transport de la lumière (voir figure 7(b)).

Calculer le transport anisotrope de la lumière même pour une forme simple comme une dalle est cependant encore très complexe [Cha60]. Pour accélérer le calcul du transport de la lumière, nous caractérisons le transfert radiatif dans une dalle à travers une *fonction de transport canonique*. Ceci est décrit en section 5. Nous obtenons cette fonction en étudiant de nombreuses simulations de Monte-Carlo pour différents paramètres de dalle (voir annexe A).

Nous implémentons notre approche de rendu complète sur le GPU, en utilisant des *depth maps* pour représenter la surface du nuage. Ceci est décrit en section 7. En section 8 nous introduisons notre enrichissement de la forme du nuage et en particulier de sa silhouette dans une Hypertexture. Le rassemblement de toutes ces étapes à l'intérieur d'un shader est synthétisé en section 8.1. Nous présentons nos résultats et performances en section 9.

5. Caractérisation du transport lumineux canonique

5.1. Simulation

Dans cette section nous décrivons notre mise en place expérimentale dans le cas canonique d'une dalle d'épaisseur t (voir figure 7(a)). Pour chaque groupe d'ordres de dispersion, nous calculons le transport lumineux T (*i.e.*, la proportion d'énergie lumineuse transmise), le centre du collecteur $\mathbf{c}$ et sa taille σ.

Ces valeurs $\langle T, \mathbf{c}, \sigma \rangle$ sont calculées en fonction de 5 paramètres d'entrées : l'épaisseur de la dalle t, la profondeur du point de vue dans la dalle d, les angles de vue (ϕ_V, ψ_V) et l'angle incident d'illumination ϕ_L, dans le repère de référence figure 7(a). Nous définissons $\mu_V = \cos(\phi_V)$, $\mu_L = \cos(\phi_L)$, $\vec{\omega}_V$ = direction de vue, $\vec{\omega}_L$ = direction d'illumination, $\cos\theta = \vec{\omega}_V \cdot \vec{\omega}_L$, $\mathbf{p} = (0, -d, 0)$ = point de vue. Étant donné que nous voulons caractériser le transport de la lumière vers un point $\mathbf{p}$ quelconque, nous prenons des valeurs de la profondeur du point de vue d à l'intérieur de la dalle ($0 \leq d \leq t$). Les points de vue hors de la dalle correspondent à $d = 0$ ou $d = t$.

Nous lançons des simulations de Monte-Carlo des chemins lumineux pour différentes valeurs de ces 5 paramètres (voir annexe A pour plus de détails). Nous stockons les résultats $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ dans une table 5D pour chaque groupe d'ordres de dispersion. Nous appelons $\langle T, \mathbf{c}, \sigma \rangle(\phi_V, \psi_V, \phi_L, d, t)$ la *fonction de transport canonique*.

5.2. Compression des résultats

Ces tables 5D (une par groupe d'ordres de dispersion) encodent la *fonction de transport canonique*, c'est-à-dire le comportement macroscopique de la lumière dans une dalle de nuage en fonction des paramètres d'entrée $(\phi_V, \psi_V, \phi_L, d, t)$. Cet ensemble de tables 5D ne peut pas être stocké en mémoire GPU. Nous les compressons en approximant les résultats calculés par des fonctions empiriques. L'annexe A décrit notre schéma de simulation et de fitting.

• Pour le transport lumineux T, cet ajustement donne

$$T = P \cdot A \cdot X \cdot \mu_L \frac{\log(d+D)}{\log(B+D)} e^{-\frac{(d-B)^2}{2C^2}}$$

avec $A = \mathbf{A}(t,\mu_V)$, $B = \mathbf{B_1}(t,\mu_V) - \mathbf{B_2}(t,\mu_L)$, $C = \mathbf{C}(t,\mu_V)$, $D = \mathbf{D}(t,\mu_V)$, $X = \mathbf{X}(t,\mu_L)$, $P = \mathbf{P}(\theta)$, *i.e.*, une table 5D pour T est réduite en une fonction analytique et sept tables 2D. **P** encode l'anisotropie du résultat et vaut 1 si le comportement est isotrope (ordres 31-∞). **X** module le résultat selon l'angle d'illumination incident. **A**, $\mathbf{B_1}$, $\mathbf{B_2}$, **C**, **D** sont les paramètres d'une fonction Gaussienne "déformée" représentant le comportement de la lumière selon la profondeur du point de vue. La figure 8 montre les résultats de cet ajustement.

• Pour le centre du collecteur **c**, notre compression résulte en $\mathbf{c} = (c_x, 0, c_z)$ avec

$$\begin{aligned} c_x &= A_x \log(1 + E \cdot d) + B_x, \\ c_z &= A_z \log(1 + E \cdot d) + B_z, \\ A_x &= F \sin(\psi_V) \sin(G \cdot \phi_L), \\ B_x &= H \sin(\psi_V) \sin(\phi_L), \\ A_z &= I + J[\cos(\psi_V) \sin(K \cdot \phi_L) + L\phi_L], \\ B_z &= M + N \cos(\psi_V) \end{aligned}$$

avec $E = \mathbf{E}(\mu_V)$, $F = \mathbf{F}(\mu_V)$, $G = \mathbf{G}(\mu_V)$, $H = \mathbf{H}(\mu_V)$, $I = \mathbf{I}(\mu_V)$, $J = \mathbf{J}(\mu_V)$, $K = \mathbf{K}(\mu_V)$, $L = \mathbf{L}(\mu_V)$, $M = \mathbf{M}(\mu_V)$, $N = \mathbf{N}(\mu_V, \mu_L)$.

• Notre compression de la taille du collecteur σ donne

$$\sigma = \mathbf{O} + \mathbf{Q} \cdot t \cdot \log(1 + \mathbf{R} \cdot d) + \mathbf{S} \cdot \log(1 + \mathbf{T} \cdot t)$$

où **O**, **Q**, **R**, **S**, **T** sont des constantes pour un ordre de dispersion donné.

6. Estimation du transport lumineux dans les nuages

6.1. Dispersion multiple

Pour un pixel donné du nuage à rendre (qui correspond à une position **p** dans le nuage), pour chaque groupe d'ordres de dispersion, nous cherchons l'aire collectrice $(\hat{\mathbf{c}}, \hat{\sigma})$, *i.e.*, l'origine des chemins lumineux dispersés sur la surface illuminée qui sont parvenus en **p** dans la direction de vue (voir figure 7(b)). Sachant qu'un nuage est un volume de densités hétérogènes, définir sa surface est difficile. Nous la définissons comme l'iso-surface où la densité ρ vaut une valeur ρ_0 définie par l'utilisateur. Comme expliqué en section 4, nous assumons que le nuage se comporte localement comme une dalle tangente à la surface à la position du collecteur. L'orientation de la surface est ici une notion dépendante de l'échelle. Étant donné que nous sommes intéressés par la surface d'entrée des chemins lumineux *dispersés*, notre orientation locale de surface est obtenue en *filtrant* la surface du nuage selon l'écart-type de l'étalement σ (forme du nuage en pointillés dans les figures 7(b) et 9). Les sections 7 et 8 donnent plus de détails sur ce filtrage.

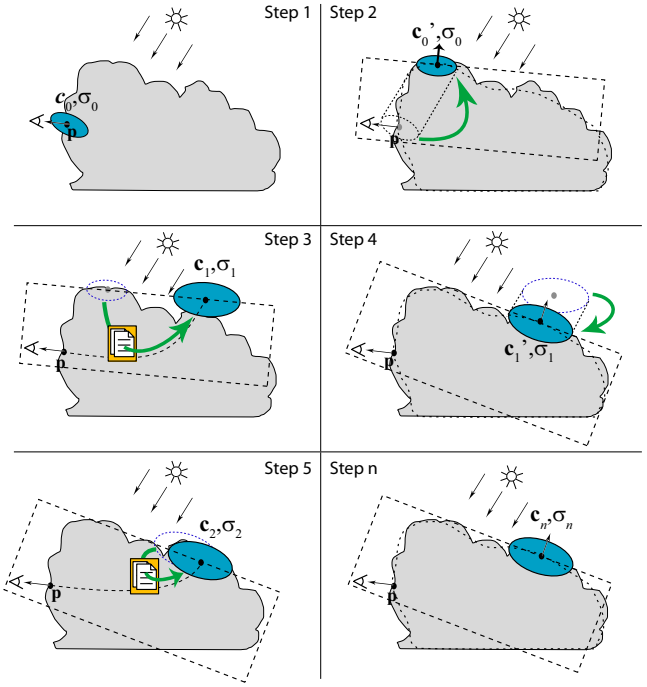


Figure 9: *Résumé de notre algorithme itératif permettant de trouver le collecteur (aire bleue). Forme de nuage solide : surface du nuage non filtrée. Forme de nuage en pointillés : surface filtrée par σ. Rectangle en pointillés : dalle tangente à la surface filtrée en* **c**. *Les symboles oranges représentent l'utilisation de notre fonction de transport canonique.*

Pour trouver la position de ce collecteur, nous itérons comme illustré en figure 9. Nous démarrons à l'étape 1 avec une taille initiale σ_0 et une position initiale $\mathbf{c}_0$ que nous projetons à l'étape 2 dans la direction de la lumière sur la surface du nuage en $\mathbf{c}'_0$. La dalle correspondante est tangente à la surface (filtrée selon σ_0) en $\mathbf{c}'_0$. Les paramètres de vue et d'illumination $(\phi_V, \psi_V, \phi_L, d, t)$ par rapport à cette dalle sont obtenus par simples transformations géométriques. A l'étape 3, la fonction de transport canonique nous donne la position du collecteur $\mathbf{c}_1(\phi_V, \psi_V, \phi_L, d, t)$ et sa taille $\sigma_1(\phi_V, \psi_V, \phi_L, d, t)$ correspondant à une telle configuration, qui est probablement différente de $\mathbf{c}'_0$. Nous projetons $\mathbf{c}_1$ sur la surface du nuage filtrée par σ_1 en étape 4, et itérons jusqu'à convergence, c'est-à-dire $\mathbf{c}_n \simeq \mathbf{c}'_n$. Nous prenons alors $(\hat{\mathbf{c}}, \hat{\sigma}) = (\mathbf{c}_n, \sigma_n)$. Une fois le collecteur trouvé, nous pouvons obtenir le transport lumineux T correspondant (*i.e.*, la quantité d'énergie transpose depuis ce collecteur jusqu'à l'œil) avec les mêmes paramètres d'entrée. Nous effectuons cet algorithme pour chaque groupe d'ordres de dispersion sur le GPU (voir section 7).

Cet algorithme itératif est en fait une méthode du point fixe appliquée sur **c**, *i.e.*, nous cherchons la valeur x qui satisfait $f(x) = x$, où x représente nos paramètres de collecteur $(\mathbf{c}, \sigma)$ et f représente notre fonction de transport canonique.

Étant donné que l'espace de recherche est restreint à la surface illuminée du nuage, cet algorithme ne peut pas diverger. Cependant, comme n'importe quel méthode du point fixe basique, il peut atteindre un état où il boucle d'une valeur à une autre sans converger (*e.g.*, $\mathbf{c}_i \neq \mathbf{c}_{i-1}$ et $\mathbf{c}_i = \mathbf{c}_{i-2}$). Il peut également prendre des pas trop grand et manquer la solution. Pour éviter ces cas, nous limitons la taille de chaque pas ($\mathbf{c}_i$ est contraint tel que $\|\mathbf{c}_i - \mathbf{c}_{i-1}\| < \sigma_{i-1}$). Nous démarrons l'algorithme avec une position initiale $\mathbf{c}_0 = \mathbf{p}$. Nous choisissons une taille initiale σ_0 grande. En effet, si le collecteur initial comprend la totalité du nuage, le résultat projeté à l'étape 2 sera une approximation au premier ordre de la surface illuminée totale, ce qui est un bon point de départ pour notre algorithme en terme de vitesse de convergence. Les étapes suivantes vont ensuite "raffiner" la surface illuminée à la position du collecteur le plus probable.

6.2. Single scattering

Pour le premier ordre de dispersion, qui constitue un cas dégénéré pour notre algorithme basé sur le collecteur, nous intégrons analytiquement la dispersion de Mie le long de la direction de vue. Ceci nous permet de prendre en compte les densités hétérogènes et les détails fin le long de la direction de vue.

Cette intégration est fait en considérant des segments successifs comme illustré en figure 10 (*i.e.*, des échantillons espacés exponentiellement sont lus le long de la direction de vue). Pour un segment i, la formule de la dispersion primaire donne

$$\begin{aligned} T_i &= \mathcal{M}ie(\theta) \int_{x_i}^{x_{i+1}} \kappa e^{-\kappa(x+l(x))} dx \\ &= \mathcal{M}ie(\theta)(e^{-\kappa(x_{i+1}+l_{i+1})} - e^{-\kappa(x_i+l_i)}) \end{aligned} \quad (1)$$

où $\mathcal{M}ie$ est la fonction de phase de Mie et κ le coefficient d'extinction décrit par l'équation 4, annexe A. À noter que puisque la fonction de phase de Mie est dépendante de la longueur d'onde, nous encodons et utilisons $\mathcal{M}ie$ comme une fonction RVB. En revanche, pour la dispersion multiple, cette dépendance est atténuée et il est suffisant de représenter T par un scalaire.

6.3. Opacité

L'opacité α est calculée en intégrant l'extinction le long de la direction de vue. Ceci résulte en $\alpha = \int_0^\infty e^{-\kappa x} dx$. Puisque le coefficient d'extinction κ varie dans l'espace, nous discrétisons dette intégrale en segments le long de la direction de vue, comme pour le *single scattering*. Pour chaque segment i, nous lisons κ dans le volume du nuage et accumulons l'opacité

$$\alpha_i = \int_{x_i}^{x_{i+1}} e^{-\kappa x} dx = \frac{e^{-\kappa x_{i+1}} - e^{-\kappa x_i}}{\kappa}. \quad (2)$$

Ceci est une procédure standard de *ray marching*.

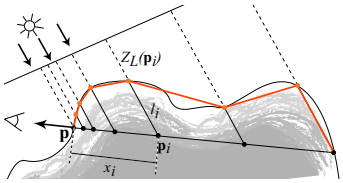


Figure 10: *Intégration linéaire par morceaux des chemins de la dispersion primaire.*

7. Implémentation sur GPU

Au moment du rendu, le maillage du nuage est rendu en utilisant un *fragment shader* qui calcule le transport lumineux depuis la surface illuminée comme expliqué en section 6. Étant donné que ce processus est lourd, nous tirons parti des capacités du GPU pour le rendre efficace, en particulier pour le calcul des distances à la surface filtrée, que nous détaillons dans cette section.

Les 22 fonctions intermédiaires **A** à **X** décrites en section 5.2 et nécessaires pour calculer le la fonction de transport canonique sont discrétisées et stockées sous forme de textures, ainsi que la fonction de phase $\mathcal{M}ie$ utilisée dans la dispersion primaire. En tout, cela donne quelques textures occupant moins de 2Mo en mémoire vidéo.

Conversion de la géométrie en *depth maps*
Pour gérer efficacement le traitement de la surface du nuage (*i.e.*, la filtrer et calculer les distances à cette surface filtrée) sur le GPU, nous nous basons sur des *depth maps*. Nous créons deux *depth maps* $Zmin_L$ et $Zmax_L$ et une *normal map* N_L depuis le point de vue de la lumière. Ceci est effectué à chaque image pour permettre l'animation de la forme du nuage et de la lumière.

Une façon pratique d'approximer le filtrage de la surface par un noyau de taille σ est de se baser sur le MIP-mapping des valeurs de profondeur Z dans la *depth map*. Pour calculer le MIP-mapping correctement (*i.e.*, sans prendre en compte les pixels de la *depth map* où il n'y a pas de nuage), nous ajoutons un canal alpha A et suivons la même procédure que pour les textures à alpha prémultiplié [DS03]. En fait cette prémultiplication est faite automatiquement lorsqu'on règle la valeur par défaut de Z à 0. Une fois que la pyramide de MIP-map $(Z^0, \ldots, Z^n)$ est calculée, une valeur Z filtrée est obtenu par $\frac{Z^i(P)}{A^i(P)}$.

Le calcul des distances d (section 6.1) et l_i (équation 1) à la surface filtrée dans la direction de la lumière consiste à simplement accéder à la *depth map* au niveau de MIP-map associé. Pour calculer l'épaisseur t, nous soustrayons $Zmax_L - Zmin_L$. L'orientation de la surface filtrée est obtenue en lisant dans la *normal map* MIP-mappée N_L. Grâce à ces représentations, tous les paramètres d'entrée $(\phi_V, \psi_V, \phi_L, d, t)$ peuvent être calculés et l'algorithme complet décrit en section 6 peut être implémenté dans un fragment shader. Pour chaque groupe d'ordres de dispersion, le shader commence par trouver le collecteur $(\hat{\mathbf{c}}, \hat{\sigma})$ avec notre

algorithme décrit en section 6.1, puis calcule l'intensité T associée en utilisant la fonction de transport canonique compressée décrite en section 5.

8. Enrichissement volumique du modèle de nuage

Les approches pour les nuages basées sur les surfaces reposent généralement sur un shader de transparence donnant l'illusion d'hétérogénéités sur la silhouette. Les méthodes basées sur des volumes sont limitées en résolution en raison de besoins en mémoire importants, ce qui en général donne une silhouette manquant de détails (voir figure 5). Notre approche combine le meilleur des deux : nous définissons une Hypertexture volumique dans une couche sous la surface du maillage afin d'avoir des effets 3D à haute résolution sur les bords du nuage en utilisant peu de mémoire (voir figure 6). Ainsi, la représentation volumique ajoute toutes les fréquences de détails du nuage qui ne sont pas représentées par la géométrie.

Cette Hypertexture dépend d'un *distance field* $D(\mathbf{p})$ à la surface (qui vaut 0 sur la surface et augmente à l'intérieur du nuage). Les détails ajoutés par ce volume sont définis procéduralement en modulant la densité N_0 (équation 3) et le coefficient d'extinction κ (équation 4) avec $\rho(\mathbf{p}) = S(D(\mathbf{p}) + noise(\mathbf{p}))$ où S est une fonction sigmoïde et *noise* un bruit de Perlin scalaire 3D [Per85]. Étant donné le coût de calcul d'un *distance field*, $D(\mathbf{p})$ est calculé et stocké dans une texture volumétrique. $noise()$ est évalué à la volée dans le shader. Nous nous basons sur [ano08] pour calculer et voxéliser rapidement le champs de distance à haute résolution en utilisant uniquement la mémoire minimum nécessaire.

Cet enrichissement volumique est utilisé en trois endroits du shader. Lorsque nous calculons l'opacité du nuage (section 6.3), nous effectuons un *ray marching* sur le GPU [ano08] à travers cette couche pour intégrer la densité du nuage. Lorsque nous calculons la dispersion simple (section 6.2), le coefficient d'extinction κ (équation 4) est également modulé pour chaque segment par $\rho(\mathbf{p})$. Enfin, pour prendre en compte ces détails dans le calcul de la dispersion multiple, (section 6.1), la *depth map* $Zmin_L$ est modulée : nous stockons dans $Zmin_L$ la profondeur du premier voxel ayant $\rho > \rho_0$.

À noter que pour la plupart des pixels hormis ceux sur la silhouette et sur les parties fines, la forte opacité arrêtera le rayon peu après le point d'entrée, la longueur moyenne du *ray marching* (et donc son coût) est donc limitée.

8.1. Rendu final

Les précalculs pour l'image courante sont limités à la reconstruction des *depth maps* et de la *normal map*. Ceci nous permet de changer aussi bien les conditions de vue et d'illumination que la forme du nuage. En cas d'animation du nuage, nous utilisons l'algorithme décrit dans [ano08] pour recalculer en temps-réel le champ de distance utilisé par notre bruit procédural.

Le terrain et les objets opaques sont dessinés en premier. Ensuite le maillage du nuage est dessiné par *deferred shading* en utilisant notre pixel shader. Ce pixel shader :

- trouve itérativement les collecteurs $(\hat{\mathbf{c}}, \hat{\sigma})$ pour chacun des 8 groupes d'ordres de dispersion grâce à l'algorithme décrit en section 6.1;
- calcule le transport lumineux T associé à chaque groupe comme expliqué en section 5, et multiplie chacun par les conditions d'illumination (intensité, couleur, visibilité);
- calcule la dispersion primaire (section 6.2);
- calcule l'opacité (section 6.3).

Nous prenons en compte l'illumination de l'environnement de la même façon que l'illumination du soleil. Une source de lumière de couleur bleue est ajoutée au-dessus du nuage et une autre de couleur marron en dessous. Nous utilisons notre algorithme de dispersion multiple avec ces deux sources additionnelles. Les scènes extérieures requièrent la prise en compte de l'épaisseur atmosphérique (*aerial perspective*). Nous utilisons le modèle de [HP03]. Le rendu d'objets très lumineux tels que les nuages demande également une gestion du *tone mapping*. Nous utilisons une version simplifiée et non floue de [GWWH03].

À noter que notre représentation nous permet implicitement de prendre en compte les points de vue à l'intérieur des nuages, puisque notre fonction de transport canonique tient compte de ce cas.

9. Résultats

Nos tests ont été conduits sur un Pentium 4 à 1.86 GHz avec une carte graphique nVidia 8800 GTS, à une résolution de 800×600.

Nous avons testé notre méthode sur différentes formes de nuages: une dalle de nuage, une couche de stratocumulus animée sans bruit procédural, et un cumulus avec bruit procédural. La dalle nous permet de tester notre algorithme et valider directement ses résultats par rapport à la fonction de transport canonique. Elle nous permet également de voir des effets tels que la réflectance anisotrope, la transmittance anisotrope dans les dalles fines et isotrope dans les dalles épaisses. La couche de stratocumulus nous permet de tester notre méthode sur une forme proche de la dalle canonique et de valider la gestion des animations. Elle est composée de 130 000 triangles et la vitesse est de 10 images par seconde. Nous pouvons voir sur cet exemple tous les effets recherchés caractéristiques des nuages : la dispersion anisotrope, la diffusion, la rétro-dispersion, la gloire, les arcs blancs. Le modèle de cumulus nous permet de tester notre algorithme sur une forme quelconque. Il est composé de 5 000 triangles et d'une Hypertexture de résolution 512^3. Nous obtenons une

vitesse de 2 images par secondes sur ce modèle. La majeure partie du temps est passée à évaluer le bruit à la volée dans l'Hypertexture. Sans ce bruit, le rendu monte à 10 images par secondes. Ce modèle présente également tous les effets recherchés, et possède les détails nécessaires au réalisme. 10 itérations de notre algorithme de recherche de collecteur sont suffisantes dans tous les cas pour converger. La figure 11 montre les résultats de notre méthode sur des cas variés.

10. Discussion et Perspectives

Nous abordons les limites des autres approches temps-réel [HL01, REK*04, HAP05] en traitant précisément les chemins lumineux de tous les ordres sur un modèle de nuage plus détaillé. Nous utilisons la fonction de phase de Mie pour les gouttelettes d'eau, qui nous donne la bonne anisotropie et nous permet de reproduire les effets visuels tels que la gloire et les arcs blancs. Au contraire de [SSEH03], nous tenons compte des détails procéduraux dans le calcul du transport lumineux. Bien que des détails similaires peuvent être obtenus par les autres techniques temps-réel [HL01, REK*04,HAP05] en augmentant la résolution volumique de leurs modèles (au détriment de la vitesse et dans les limites de la mémoire disponible), notre méthode apporte des effets qui ne sont pas pris en compte par ces approches tels que la rétro-dispersion et la diffusion. Nous permettons l'animation du point de vue, de la lumière et de la forme du nuage.

Comme pour les autres méthodes utilisant une *depth map* pour résoudre des problèmes de dispersion [DS03], cette représentation nous pose quelques problèmes vis-à-vis de la résolution et des formes non convexes. La précision de cette représentation est limitée par la résolution de la *depth map*. En conséquence, le passage à l'échelle de la méthode requièrerait probablement des techniques de *depth maps* alternatives [SD02, Arv07]. Dans le cas de formes non convexes, le transport lumineux est légèrement sous-estimé dans les parties ombrées (voir figure 12(a)) : l'algorithme assume qu'il y a de la matière entre les surfaces à l'ombre et la surface éclairée, ce qui est incorrect. Il en résulte que la lumière est considérée plus atténuée que ce qu'il faudrait. En pratique, cette erreur survient dans les parties ombragées, où l'illumination par l'environnement (*i.e.*, par le ciel et le sol) est prédominante. Comme indiqué dans [DS03], qui souffrent des même problèmes, ce point peut être résolu en utilisant une technique de *depth peeling* pour traiter les formes non convexes comme une collection de formes convexes. L'utilisation de *deep shadow maps* [LV00] aiderait également à résoudre ce problème et augmenterait la précision de notre calcul de dispersion primaire.

Une limitation de notre approche est la supposition que la lumière arrivant à l'utilisateur passe par seulement un seul collecteur connexe (*i.e.*, un seul chemin le plus probable) par groupe d'ordres de dispersion. Comme indiqué sur la figure 12(b), cela n'est pas toujours le cas. Dans ces configurations, notre algorithme ne prend en compte qu'un collecteur, et sous-estime donc la quantité de lumière transmise. Cependant, puisque nous cherchons plusieurs collecteurs par pixel (un pour chacun des 8 groupes d'ordres de dispersion), cette erreur ne concerne qu'une fraction de la couleur d'un pixel. Une solution pourrait être de chercher plusieurs collecteurs par groupe d'ordres de dispersion, avec différentes valeurs initiales.

Rechercher un collecteur $\hat{\mathbf{c}}$ correspond à chercher le point d'entrée $\hat{\mathbf{c}}$ sur la surface illuminée d'un maximum local du transport de la lumière T. Notre algorithme itératif correspond à une méthode du point fixe : nous cherchons le point fixe $f(\hat{\mathbf{c}}) = \hat{\mathbf{c}}$ où f représente notre fonction de transport canonique. Dans le futur, de meilleures techniques du milieu de l'optimisation pourraient être utilisées pour garantir une convergence plus rapide et pour traiter le cas de plusieurs maxima de T.

Notre calcul du transport lumineux sur GPU utilisant notre algorithme de recherche de collecteur (*i.e.*, la principale contribution de cet article) est suffisamment rapide pour des applications temps-réel (10 images par seconde). La rapidité de cette méthode peut encore être augmentée. De plus, le calcul du bruit procédural et le *ray marching* sur GPU dans notre implémentation sont des versions non optimisées de [ano08] et prennent un temps de calcul conséquent (80% du coût de rendu).

Dans le futur, nous aimerions prendre en compte les effets lumineux de plus haut niveau tels que les inter-réflexions entre les nuages et entre les lobes des nuages. Aussi, la méthode décrite dans [BNL06] peut être appliquée à notre modèle pour calculer les inter-réflexions entre les nuages et le sol, qui ont une certaine importance visuelle.

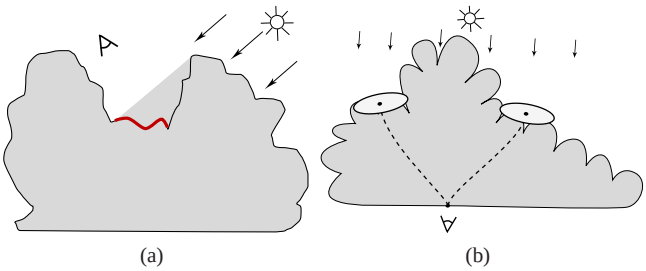


Figure 12: ***(a)*** *Limites de l'utilisations de depth maps. L'illuminance de la surface schématisée en rouge est légèrement sous-estimée. Cependant, l'erreur est de faible importance visuelle.* ***(b)*** *Limites de la recherche d'un seul collecteur par groupe d'ordres de dispersion. Dans cet exemple, deux collecteurs d'importances équivalentes existent, mais notre méthode n'en cherchera qu'un.*

11. Conclusion

Nous avons présenté une étude du transport lumineux débouchant sur une nouvelle formulation du comportement macroscopique de la lumière dans les milieux participants adaptée à la synthèse d'image. Nous avons présenté une nouvelle façon de calculer le transfert radiatif à travers un vol-

ume homogène de milieu participant avec une méthode optionnelle dédiée au traitement des bordures inhomogènes. Contrairement aux approches précédentes, notre méthode n'a besoin que de parcourir les bordures du milieu et non pas le volume entier. Nous avons démontré la validité de cette technique sur des nuages détaillés de type cumulus. Comme on peut le voir dans nos résultats, nous reproduisons correctement les effets visuels tels que la rétro-dispersion, la dispersion anisotrope multiple, la gloire, etc.

12. Remerciements

De très gros mercis vont à Guillaume Piolat pour les jours et les nuits passés sur le code et à Laurence Boissieux et Florent Moulin pour les week-ends passés à modéliser les nuages. Merci également à Sébastien Barbier et Paul Kry pour les relectures. Merci enfin à Frédéric Saint-Marcel et Eric Ragusi pour l'utilisation et la maintenance du cluster et de la base de données. Le LLNL a également mis ses clusters à disposition. Les fonctions de Mie ont été générées grâce au précieux outil MiePlot de Philip Laven.

Appendix A: Mesurer le transport lumineux dans une dalle de nuage

Comme expliqué en section 2, la fonction de phase appropriée pour les nuages est la fonction de Mie. C'est une fonction complexe et oscillante dépendant de la taille de la gouttelette et de la longueur d'onde de la lumière. Comme [BNL06], nous pré-intégrons l'effet de la DSD sur la fonction de phase et utilisons l'approximation *Mie modifiée* dans laquelle le pic en avant (5°) est supprimé de la fonction de phase et traduit comme une densité diminuée de 50% [Len85]. Cette approche a été montrée comme donnant des résultats équivalent tout en nécessitant de traiter deux fois moins d'événements de dispersion.

Pour la DSD, nous utilisons la distribution Gamma modifiée [Lev58] pour les gouttelettes de nuages définie par

$$N(r) = \frac{\rho N_0}{\Gamma(\gamma) r_n} \left(\frac{r}{r_n}\right)^{\gamma-1} e^{-\frac{r}{r_n}}. \qquad (3)$$

Cette fonction décrit la densité $N(r)$ des gouttelettes de rayon r, avec r_n le rayon caractéristique de la distribution, γ la largeur de la distribution, et N_0 la densité totale de gouttelettes. Γ est la fonction Gamma. ρ est un facteur de modulation de la densité totale apporté par notre enrichissement procédural décrit en section 8. Une distribution de gouttelettes pour nuage est ainsi décrite par les seuls trois paramètres N_0, r_n et γ. En terme de propriétés optiques, le rayon efficace correspondant à cette distribution est $r_e = (\gamma+2) r_n$. Nous prenons comme paramètres typiques pour un cumulus $r_e = 6\ \mu$m, $\gamma = 2$, $N_0 = 4.10^8 \mathrm{m}^{-3}$.

Le coefficient d'extinction κ est défini par

$$\kappa = \rho N_0 \pi r_e^2. \qquad (4)$$

Il donne la fonction d'extinction $e^{-\kappa x}$ qui est la probabilité de traverser le nuage le long d'un segment de longueur x sans toucher une gouttelette. Il est utilisé dans toutes les équations de la dispersion, c'est-à-dire dans l'opacité (équation 2), le *single scattering* (équation 1) et la dispersion multiple (équation 5).

L'équation de la dispersion multiple a été présentée maintes fois dans la littérature [Cha60, Kv84]. Elle peut être écrite sous la forme

$$-\frac{\vec{\omega}}{\kappa} \cdot \frac{dT(\mathbf{p},\vec{\omega})}{d\mathbf{p}} = T(\mathbf{p},\vec{\omega}) + \frac{1}{4\pi} \int_{\vec{\omega}'} P(\vec{\omega}\cdot\vec{\omega}') T(\mathbf{p},\vec{\omega}') d\vec{\omega}' \qquad (5)$$

pour l'intensité T parvenant à un point $\mathbf{p}$ dans la direction $\vec{\omega}$, avec $P(\theta)$ la fonction de phase du milieu (ici, $\mathcal{M}ie$). Cette équation intégro-différentielle peut être résolue numériquement par une simulation de Monte-Carlo. La contribution de chaque ordre dans l'intensité T, ainsi que les informations du collecteur $(\mathbf{c},\sigma)$ peuvent être aisément récupérées pendant cette intégration. Pour notre caractérisation du transport lumineux (section 5), nous avons calculé la solution de l'équation 5 dans une dalle de nuage pour des valeurs variables de $(\phi_V, \psi_V, \phi_L, d, t)$. Nous avons stocké la contribution T de chaque ordre de dispersion dans une base de données, ainsi que les données de collecteurs $(\mathbf{c},\sigma)$. Ces calculs ont été fait avec un intervalle de confiance de 5% au niveau 95%. Nous les avons lancées pour 10 millions de valeurs différentes de $(\phi_V, \psi_V, \phi_L, d, t)$, résultant en une taille de base de donnée brute d'environ 25 Go. Ces calculs ont pris plusieurs semaines sur un cluster de 100 nœuds d'Itanium-2 dual-core à 900MHz. La base de données et les résultats de fitting seront disponibles en ligne.

L'analyse de ces résultats (*i.e.,* trouver des fonctions de plus basse dimension reproduisant les résultats) a été faite empiriquement en traçant $\langle T, \mathbf{c}, \sigma \rangle$ selon les paramètres d'entrées et en recherchant des comportements remarquables. Elle a été inspirée par des approches précédentes telles que les fonctions X et Y de Chandrasekhar. L'ajustement lui-même a été fait par optimisation standard non linéaire au sens des moindres carrés avec MATLAB.

References

[ano08] ANONYMOUS: Real-time rendering of large detailled volumes. In *submitted to the ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)* (2008).

[Arv07] ARVO J.: Alias-free shadow maps using graphics hardware. *journal of graphics tools 12*, 1 (2007), 47–59.

[Bli82] BLINN J. F.: Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH'82* (1982), pp. 21–29.

[BNL06] BOUTHORS A., NEYRET F., LEFEBVRE S.: Real-time realistic illumination and shading of stratiform

clouds. In *Eurographics Workshop on Natural Phenomena* (sep 2006).

[Cha60] CHANDRASEKHAR S.: *Radiative transfer*. New York: Dover, 1960, 1960.

[DKY*00] DOBASHI Y., KANEDA K., YAMASHITA H., OKITA T., NISHITA T.: A simple, efficient method for realistic animation of clouds. In *SIGGRAPH'00* (July 2000), pp. 19–28.

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Eurographics Workshop on Rendering (EGWR)* (2003), pp. 197–201.

[Ebe97] EBERT D. S.: A cloud is born. In *SIGGRAPH'97* (1997), p. 245.

[ES00] ELINAS P., STÜRZLINGER W.: Real-time rendering of 3D clouds. *J. Graph. Tools 5*, 4 (2000), 33–45.

[Gar85] GARDNER G. Y.: Visual simulation of clouds. In *SIGGRAPH'85* (1985), ACM Press, pp. 297–304.

[GWWH03] GOODNIGHT N., WANG R., WOOLLEY C., HUMPHREYS G.: Interactive time-dependent tone mapping using programmable graphics hardware. In *Eurographics Workshop on Rendering (EGRW)* (2003), pp. 26–37.

[HAP05] HEGEMAN K., ASHIKHMIN M., PREMOŽE S.: A lighting model for general participating media. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)* (2005), pp. 117–124.

[HL01] HARRIS M. J., LASTRA A.: Real-time cloud rendering. *Computer Graphics Forum 20*, 3 (2001), 76–84.

[HP03] HOFFMAN N., PREETHAM A. J.: *in Graphics programming methods*. Charles River Media, Inc., 2003, ch. Real-time light-atmosphere interactions for outdoor scenes, pp. 337–352.

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *SIGGRAPH'01* (2001), pp. 511–518.

[KE86] KRIM M. S. A., EL WAKIL S. A.: Angular distribution of radiation in an isotropically scattering slab. *Astrophysics and Space Science 121* (Apr. 1986), 137–145.

[Kv84] KAJIYA J. T., VON HERZEN B. P.: Ray tracing volume densities. In *SIGGRAPH'84* (1984), pp. 165–174.

[Len85] LENOBLE J.: *Radiative transfer in scattering and absorbing atmospheres: standard computational procedures*. A. Deepak Publishing, 1985.

[Lev58] LEVIN L. M.: Functions to represent drop size distribution in clouds. the optical density of clouds. *Izv. Akad. Nauk. SSSR, Ser. Geofiz. 10* (1958), 198–702.

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *SIGGRAPH'00* (2000).

[Max94] MAX N. L.: Efficient light propagation for multiple anisotropic volume scattering. In *Eurographics Workshop on Rendering (EGWR)* (1994), pp. 87–104.

[MMKW97] MAX N. L., MOBLEY D., KEATING B., WU E.: Plane-parallel radiance transport for global illumination in vegetation. In *Eurographics Workshop on Rendering (EGWR)* (1997), pp. 239–250.

[Mob89] MOBLEY C.: A numerical model for the computation of radiance distributions in natural waters with wind-roughened surfaces. *Limnol. Oceanogr. 34* (1989), 1473–1483.

[Ney03] NEYRET F.: Advected textures. In *ACM SIGGRAPH/EG Symposium on Computer Animation (SCA)* (july 2003).

[NND96] NISHITA T., NAKAMAE E., DOBASHI Y.: Display of clouds taking into account multiple anisotropic scattering and sky light. In *SIGGRAPH'96* (Aug. 1996), pp. 379–386.

[PAS03] PREMOŽE S., ASHIKHMIN M., SHIRLEY P.: Path integration for light transport in volumes. In *Eurographics Symposium on Rendering (EGSR)* (2003), pp. 52–63.

[PAT*04] PREMOZE S., ASHIKHMIN M., TESSENDORF J., RAMAMOORTHI R., NAYAR S.: Practical rendering of multiple scattering effects in participating media. In *Eurographics Symposium on Rendering (EGSR)* (June 2004), pp. 363–374.

[Per85] PERLIN K.: An image synthesizer. In *SIGGRAPH'85* (July 1985), pp. 287–296.

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. In *SIGGRAPH'89* (July 1989), pp. 253–262.

[REK*04] RILEY K., EBERT D. S., KRAUS M., TESSENDORF J., HANSEN C.: Efficient rendering of atmospheric phenomena. In *Eurographics Symposium on Rendering (EGSR)* (June 2004), pp. 375–386.

[SD02] STAMMINGER M., DRETTAKIS G.: Perspective shadow maps. In *SIGGRAPH'02* (2002).

[SSEH03] SCHPOK J., SIMONS J., EBERT D. S., HANSEN C.: A real-time cloud modeling, rendering, and animation system. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)* (2003), pp. 160–166.

[Sta95] STAM J.: Multiple Scattering as a Diffusion Process. In *Eurographics Workshop on Rendering (EGWR)* (1995), pp. 41–50.

[TB02] TREMBILSKI A., BROSSLER A.: Surface-based efficient cloud visualisation for animation applications. In *WSCG* (2002), pp. 453–460.

Figure 11: *Quelques résultats de notre méthode.*

Journées de l'Association Francophone d'Informatique Graphique, Marne la Vallée, 2007

Compression progressive de modèles de plantes à base de cylindres généralisés

S. Mondet[1], F. Boudon[2a], J.C. Hoelt[1], G. Morin[1], R. Grigoras[1], C. Pradal[2b], M. Paulin[1]

[1]Institut de Recherche en Informatique de Toulouse, Université de Toulouse
[2] [a]INRIA, [b]CIRAD, Virtual Plants Team, UMR DAP, TA A-96/02, Avenue Agropolis, 34398 Montpellier Cedex 5, France

Abstract/Résumé

*This paper presents our recent work on progressive compression of plant models based on generalized cylinders. This multi-scale representation is compatible with a direct-acyclic graph representation that allows us to build upon the progressive streaming work of [COM*07]. We present a method for differential coding of plants : an average branch is computed for any chosen group of branches and then, for each branch, we only need to code a transformation and differences. To be able to stream, we identify and take advantage of two types of dependencies : topological (between mother and daughter branches) and dependencies due to differential coding. We obtain a progressive model that makes it possible to select a lightweight representation of a plant while preserving branch density.*

*Ce papier présente nos travaux récents sur la compression progressive de modèles de plantes à base de cylindres généralisés. Cette représentation multi-resolution est compatible avec une représentation sous forme de graphe orienté sans cycle, ce qui nous permet de bénéficier des techniques de streaming progressif proposées dans [COM*07]. Un codage différentiel des plantes est présenté : pour un groupe choisi de branches on calcule une branche moyenne, et pour chaque branche, il reste à coder une transformation et des différences. En vue du* streaming*, nous identifions et exploitons deux types de dépendances : topologiques (entre branche fille et mère) et dues au codage différentiel. Nous obtenons un modèle progressif qui permet la sélection d'une représentation plus légère de la plante, tout en gardant la même densité de branches.*

1. Introduction

Les dernières avancées des technologies 3D (scanner, reconstruction, etc.) et des réseaux ouvrent de multiples perspectives en terme d'applications virtuelles, partagées et/ou collaboratives : exploration de scènes 3D, visites virtuelles, aménagement de paysages etc. Par exemple, le projet Michelangelo de l'Université de Stanford, qui a pour but l'archivage 3D du patrimoine, s'est intéressé à la représentation 3D de statues et a proposé un logiciel, ScanView, pour permettre aux utilisateurs un accès à distance à ces statues. Second Life, un monde virtuel partagé très populaire, encourage ses utilisateurs à créer et échanger des objets 3D. Si actuellement Second Life ne permet que la création d'objets 3D à géométrie solide, les utilisateurs demandent davantage de flexibilité avec l'intégration d'objets plus complexes.

Dans ce contexte, la simulation de scènes naturelles virtuelles est un challenge important pour la représentation de mondes virtuels à des fins scientifiques (architecture, aménagement du territoire, modélisation écologique) ou plus ludiques (jeux vidéo, cinéma).

Le contexte applicatif de ce travail est donc de permettre la mise en ligne de vastes scènes 3D naturelles et réalistes. Une navigation à travers cette scène est offerte à un client à distance. Un téléchargement complet de la scène doit être évité pour diminuer la latence du téléchargement et permettre la navigation à partir d'un client relativement léger. L'idée est de transmettre la scène progressivement, et de

pouvoir visualiser les objets au fur et à mesure de leur réception. L'ordre des objets est décidé relativement à la navigation dans la scène (par exemple, adaptation au point de vue). Certains objets, notamment des plantes, peuvent être volumineux (plusieurs dizaines de Mo), et nous souhaitons donc pouvoir, au niveau d'un seul objet, avoir une représentation progressive. Une telle représentation permet de commencer à visualiser une version simplifiée de l'objet, en attendant que la totalité du modèle soit parvenue au client. Cette représentation progressive, comme les représentations progressives de vidéo, introduisent de la dépendance entre les données. Une donnée ne peut être décodée que lorsque les données dont elle dépend ont déjà été reçues. Par exemple, dans le contexte de vidéo progressive de type MPEG, une image ne peut être décodée que quand l'image de référence dont elle dépend est reçue. Cependant, les données vidéo (ou audio) sont par nature très différentes des données 3D. En effet, les données vidéo ont des contraintes temporelles très fortes : les résultats sur le *streaming* vidéo ne sont donc pas applicables à notre problématique. Une méthode de *streaming* de contenu 3D a été proposée dans [COM*07]. Nous souhaitons utiliser cette méthode pour le streaming de plantes : en proposant un organisation des données compatible avec cette méthode, la mise en paquet, et l'ordonnancement de ces paquets sont déterminés. Pour cela, une représentation progressive adaptée aux plantes et cohérente pour l'approche de *streaming* est proposée. Le modèle original est une représentation d'un arbre par cylindres généralisés. Dans ce papier, un codage différentiel de cette plante est développé duquel découle une représentation progressive.

2. Contexte du travail

2.1. Le projet Natsim

Ce travail s'inscrit dans le cadre du projet ANR *NatSim* [nat05]. Ce projet concerne la définition de structures de données, d'algorithmes et de métaphores pour la simulation, la modélisation, la visualisation et la transmission de scènes naturelles. Afin de permettre une gestion efficace de la complexité inhérente aux scènes naturelles, le coeur du projet repose sur un ensemble de représentations multi-modèles, multi-échelles et multi-résolutions des composants d'une scène naturelle. Ce projet collaboratif et pluridisciplinaire aborde la problématique de la simulation de scènes naturelles à travers cinq *aspects* :

- une structure de données compacte, souple et extensible ;
- les méthodes et processus pour l'acquisition, l'édition et la modélisation ;
- le rendu rendu temps-réel ;
- l'animation et la simulation ;
- l'échange de données par streaming adaptatif.

2.2. Plateforme expérimentale

Afin de permettre l'expérimentation et la recherche dans le domaine de la modélisation par esquisse, du rendu temps-réel, de la simulation de paysage ainsi que des méthodes de travail collaboratif distant en environnement hétérogène, l'IRIT a développé une plateforme expérimentale de visualisation de contenu 3D. L'architecture logicielle ouverte et extensible de cette plateforme expérimentale a été définie afin que les différents partenaires du projet puissent y intégrer leurs travaux en toute simplicité. Ainsi, nos travaux ont été menés afin de rajouter à cette plateforme la possibilité de naviguer à distance dans la scène naturelle.

2.3. Intégration du streaming

Le scénario d'utilisation envisagé pour valider l'apport de notre méthode de mise en paquet pour le streaming de scènes 3D sera le suivant :

Un serveur héberge une scène 3D complexe. Un client/utilisateur à distance souhaite naviguer interactivement dans la scène. Le téléchargement complet du contenu étant hors de propos (contenu beaucoup trop volumineux - pouvant être compté en centaines de Go), le serveur se doit d'adapter (et donc de sélectionner) le contenu au point de vue de l'utilisateur ainsi qu'à l'état de ses ressources (e.g. bande passante) et de le transmettre de manière progressive afin que le client puisse visualiser une représentation (même grossière) de la scène le plus vite possible.

Pour cela nous avons développé la base d'un serveur et d'un client fondés sur la plateforme *NatSim*, en respectant son architecture modulaire.

Le présent travail se concentre particulièrement sur l'optimisation de la transmission progressive des principaux objets des scènes naturelles : les plantes.

3. Travail proposé

Nous nous intéressons donc à la transmission progressive d'objets 3D compressés pour une scène naturelle, plus précisément des plantes. Pour des objets 3D « classiques » il existe plusieurs algorithmes de compression multi-résolutions, que ceux-ci soient représentés par des maillages (e.g. *Progressive Meshes* [KLK04]) ou par des points (c.f. [KB04]).

Cependant, une représentation de plantes par maillages progressifs ne donne pas des résultats satisfaisants [RCB*02] : la topologie d'une plante est telle qu'il est difficile de supprimer des triangles à partir d'un certain niveau. Cette architecture particulière implique d'utiliser des représentations adaptées aux plantes.

Les travaux proposés ici s'appuient sur des représentations par cylindres généralisés [Blo85]. Un premier travail de simplification de cette représentation a été proposé

dans [Bou04], fondé d'une part sur l'importance de chaque branche, et sur la simplification de chaque cylindre généralisé d'autre part.

Ainsi, nous nous proposons de rechercher une méthode de compression progressive de modèles de plantes représentés par des cylindres généralisés et d'exprimer les dépendances internes à ces données afin de les transmettre efficacement. Dans le reste de cet article, nous présentons d'abord brièvement un modèle de mise en paquets (section 4), puis nous explicitons dans les sections suivantes la représentation des plantes proposée.

4. Mise en paquets pour le streaming 3D

La mise en paquets pour le streaming 3D a été assez peu étudiée. Par contre, beaucoup de travaux se sont intéressés à la compression 3D et aux représentations échelonables. Un *état de l'art* a été rédigé par Taubin [Tau99] sur la compression et la transmission progressive de géométrie 3D définie par maillages. Un premier groupe de travaux s'est intéressé à la compression non progressive. Pour la compression progressive, les maillages progressifs sont les plus utilisés. D'autres structures dérivées des maillages progressifs ont été proposées dans *Progressive Forest Split* et *Progressive Simplicial Complexes*.

Les travaux [PKL06, YKK05] minimisent les dépendances entre différentes parties du maillage, de façon à pouvoir afficher une partie de l'objet indépendamment du reste. Gu et Ooi [GO05] ont été les premiers à s'intéresser à la mise en paquets des maillages progressifs. Puis, dans le but de transmettre efficacement des objets 3D multi-résolutions, [COM*07] optimise la qualité visuelle d'un objet partiellement transmis à l'instant t, en considérant à la fois les dépendances entre les données, et leurs importances relatives. Dans ce modèle, considérer les dépendances entre les fragments de données permet de minimiser la présence sur le client de données inutilisables dues aux pertes. Les latences des retransmissions induites par le lien réseau (ici UDP + retransmission) sont prises en compte. Une évaluation statistique avant la transmission tend à maximiser la quantité d'information utilisable pour une bande passante donnée, étant donné les retards dus aux pertes de certains paquets.

Ainsi, l'algorithme de mise en paquets induit par ce modèle permet une visualisation au fil de l'eau. Les données transmises sont visualisées par le client, d'abord avec un faible niveau de détail, puis, au fur et à mesure que de nouvelles données sont disponibles, l'aspect de l'objet visualisé se précise, jusqu'à atteindre sa résolution maximum.

La clé pour pouvoir optimiser le transfert de données est de connaître les dépendances entre les différents niveaux de résolution. Ces dépendances peuvent être modélisées par un graphe orienté sans cycle (*DAG - direct acyclic graph*). De plus, à chaque noeud du graphe correspondant à un fragment des données -ou, à un raffinage de la géométrie- est associé un poids indiquant la contribution de ce fragment au rendu final.

En entrée de l'algorithme de mise en paquets, les objets 3D de la scène doivent avoir une représentation progressive. La représentation utilisée a été celle des maillages progressifs (*progressive meshes* [Hop96]). En effet, il est classique de représenter les objets 3D par des triangulations, et la transformation en maillage progressif est classique et directe. A chaque simplification (*vertex collapse* ou *edge collapse*) la distance du sommet évincé représente une mesure de la contribution de ce sommet à la géométrie. De plus, cette simplification s'appuie sur les éléments -sommets ou arêtes- voisins topologiquement du sommet simplifié. Ces dépendances correspondent donc aux dépendances du *DAG* nécessaire à l'algorithme de mise en paquets et d'ordonnancement.

Néanmoins, à cause de leur topologie ramifiée et leur géométrie particulièrement éparse, les plantes ne se prêtent pas à un codage progressif par maillage : les simplifications sur les triangles peuvent créer des artefacts (c.f. [BMG06]). Ici, nous souhaitons donc proposer une représentation progressive adaptée aux plantes qui puissent satisfaire les deux contraintes en entrée de l'algorithme de *streaming* : les dépendances entre données doivent être modélisées par un *DAG*, et un poids doit être associé à chaque noeud.

5. Représentation progressive des plantes

5.1. Représentation comme cylindres généralisés

Dans ce travail, les plantes considérées « sans leurs feuilles » sont représentées par un ensemble connecté de cylindres généralisés [Blo85]. Dans cette première version de notre travail, nous considérons que les courbes axiales générant les cylindres sont des courbes de Bézier de degré d, et que la fonction rayon est, elle aussi, une fonction de même degré. L'ensemble des cylindres généralisés est organisé sous forme d'une structure de données arborescente n-aire. Nous utilisons par la suite le terme de *n-arbre* pour la structure de données, de façon à éviter les confusions avec l'objet concret « arbre » représenté. Dans cette structure, la racine du *n-arbre* est le tronc de la plante et les branches portées par le tronc sont des filles de ce tronc.

Chaque branche fille contient une position normalisée entre $[0, 1]$ sur sa branche porteuse [PMKL01]. Ce paramètre d'attache u sur la courbe de la branche mère définie le premier point de contrôle de la courbe de Bézier de la branche fille. Les d points de contrôle restants sont codés dans la branche fille par leur trois coordonnées dans l'espace. (c.f. 5).

5.2. L'algorithme de compression des plantes

5.2.1. Codage différentiel d'une plante

Pour optimiser la compression, nous voulons exploiter la similarité des branches. L'idée de l'algorithme de compression est de remplacer le codage absolu d'une branche -les points de contrôle et poids qui la définissent- par une différence par rapport à une courbe de Bézier moyenne, pour un ensemble de branches données. Plus les branches seront similaires, plus il sera possible de quantifier cette différence sur peu de bits et donc de proposer un codage compact. Une première étape consiste donc à regrouper les branches en fonction de leur similarité.

Ainsi, les branches sont d'abord partitionnées en groupe de telle manière que la variabilité géométrique inter-groupe soit minimale pour permettre une quantification avec une perte minimum. Une première solution consiste à utiliser pour cela des algorithmes de partitionnement basés sur des comparaisons inter-branches prenant en compte le nombre de points de contrôle et les variations entre points de contrôle. Dans une première version de ce travail, nous avons discriminé les branches en fonction de leur nombre de points de contrôle.

Pour pouvoir comparer et ensuite coder de manière différentielle les branches, il est nécessaire de normaliser les cylindres généralisés les représentant. Une transformation permettant de passer de la forme normalisée à la forme réelle dans l'espace et inversement sera calculée pour chaque branche. Les représentations normalisées seront utilisées pour calculer les branches moyennes et exprimer les différences de chaque branches à leurs branches moyennes de référence sous forme de delta de points de contrôle. Nous avons choisi de transformer les cylindres afin que le premier point de contrôle P_0 de la courbe de Bézier génératrice coïncide avec l'origine et le dernier P_N avec le point $(0,0,1)$. Cette première transformation définit deux angles de rotation, et un facteur d'échelle (nous avons choisi d'utiliser une mise à l'échelle uniforme). Il reste ensuite un degré de liberté, qui correspond à la rotation autour de l'axe $\vec{z}$. Pour cela, nous choisissons l'orientation autour de l'axe $\vec{z}$ en ramenant le centre de gravité $\overline{P_i}$ des points de contrôle restants dans le plan xz.

Les transformations T résultantes de cette normalisation sont une translation $t = -\overrightarrow{P_0}$, une mise à l'échelle $s = \frac{1}{\|\overrightarrow{P_0P_N}\|}$, et 3 angles de rotation tel que $\overrightarrow{T(P_0)T(P_N)} = \vec{z}$ et $\overrightarrow{T(P_0)T(\overline{P_i})} \cdot \vec{y} = 0$. Les transformations inverses seront conservées pour chaque branche et permettent d'instancier une forme normalisée dans l'espace 3D.

Lorsque toutes les courbes génératrices des branches d'un groupe ont été normalisées, la courbe moyenne est calculée simplement telle que ses points de contrôle $\overline{P_i}$ soient les barycentres des points de contrôle P_i des n courbes.

Pour chaque branche, on pourra ensuite redéfinir les courbes de manière différentielle en fonction des courbes moyennes en stockant les coordonnées des points de contrôle sous la forme de différences aux points moyens. Il est ainsi possible de coder ces différences sur un nombre restreint de bits. On notera que les courbes étant normalisées, le premier et dernier points de contrôle n'ont pas besoin d'être codés. Pour des courbes de Bézier de degré 3 par exemple, seuls les deux points intermédiaires ont besoin d'être définis. Ainsi, le codage d'une branche est maintenant défini par des paramètres d'instanciation (une transformation) et une forme différentielle de courbe normalisée. La translation de la transformation est donnée par un scalaire définissant la position sur la branche porteuse (donnant auparavant le premier point de contrôle), trois scalaires pour les angles de rotation et un scalaire pour le changement d'échelle uniforme. Aussi, en plus du pointeur sur la branche mère qui était déjà présent dans la représentation originale, un pointeur supplémentaire vers le modèle est nécessaire.

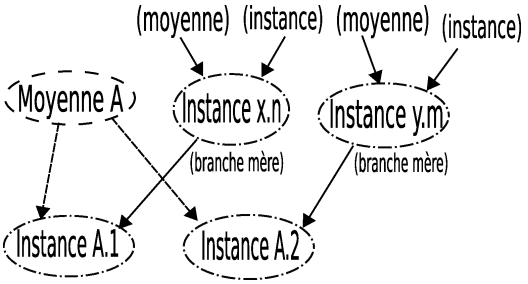


Figure 1: *Un exemple d'un certain niveau dans le DAG : par exemple l'instance notée A.1 dépend du modèle de branche A (codage différentiel) et de l'instance x.n (structure topologique de la plante).*

5.2.2. Expression des dépendances

Pour utiliser le codage différentiel d'une plante comme une représentation progressive de la plante en vue de la transmission pour une visualisation au fil de l'eau, il est nécessaire d'exprimer les dépendances et d'affecter un poids à chaque raffinage. Deux familles de dépendances existent : les premières sont les dépendances dues à la structure topologique (*n-arbre*) de la plante. C'est à dire, la dépendance entre une branche fille et la branche mère à laquelle elle est attachée. La deuxième famille correspond aux dépendances dues au codage différentiel. Une branche dépend de la branche moyenne. La figure 1 montre sur un niveau les deux formes de dépendances.

Pour pouvoir représenter la plante complète sans pour autant connaître toutes les branches en résolution maximum, il est nécessaire qu'une branche fille ne dépende pas de la version 'finale' (contenant tous les détails originaux) de sa mère. Pour cela, nous décomposons une branche donnée en niveaux de détails et déterminons la résolution nécessaire à

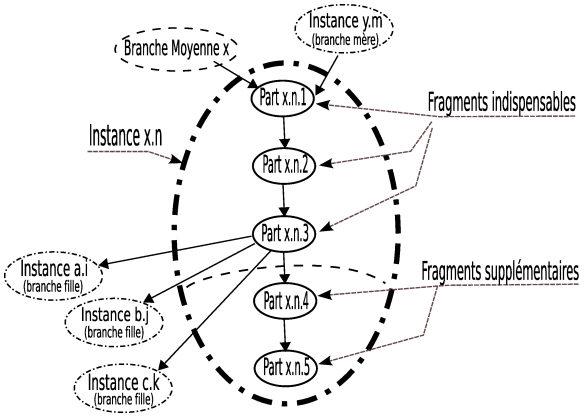


Figure 2: *Un exemple d'instance étendue : le noeud correspondant à une branche est décomposé en sous noeud qui contiennent soit des informations de transformation, soit des informations de raffinage géométrique. De cette façon, on peut exprimer un nombre de niveaux de détail arbitraire pour une branche.*

l'attache d'une branche fille. Par exemple, dans la figure 2 les branches filles de la branche $x.n$ peuvent être visualisées dès que les niveaux de détails $x.n.1$, $x.n.2$, et $x.n.3$ ont été reçus.

Ces dépendances sont forcément non cycliques ; cette structure de données peut donc effectivement être traitée par l'algorithme vu dans la section 4.

5.2.3. Évaluation de l'importance des informations

En plus de l'expression des dépendances, pour pouvoir décider de la mise en paquets et de l'ordonnancement de l'envoi de ces paquets, il faut aussi quantifier la contribution des informations fournies par chaque noeud du *DAG* à la reconstruction de la scène.

Dans notre implémentation, nous avons pour chaque branche une représentation en niveaux de détails (telle que le montre la figure 2) à deux niveaux. Le premier niveau correspond à l'instanciation de la branche moyenne, c'est à dire, codant la transformation complète. Le deuxième niveau correspond à l'introduction des différences à la branche moyenne.

Il parait d'abord naturel de privilégier les paramètres d'instanciation pour qu'au minimum toutes les branches soient représentées comme instance d'une branche moyenne, en omettant au départ le détail contenu dans le codage différentiel. De façon à favoriser d'abord l'introduction de nouvelles branches, nous ne donnons qu'un poids faible aux noeuds stockant les paramètres différentiels par rapport aux noeuds stockant les paramètres d'instanciation mais proportionnel au poids de la branche pour qu'une fois que toutes les branches soient représentées, les branches les plus importantes soient raffinées en premier.

Pour ordonner les branches, nous avons choisi, dans une première version, d'utiliser la structure hiérarchique de la plante. Nous donnons à une branche une importance proportionnelle au nombre de ses branches descendantes. Les branches terminales -feuilles du *n-arbre-* ont par défaut le poids 1, et chaque branche intermédiaire reçoit la somme des poids des branches qu'elle porte.

Finalement pour ordonner les informations des branches de même importance dans la hiérarchie, une idée est prendre en compte leur importance visuelle. Pour cela, une heuristique basée sur le volume des branches est proposée dans [Bou04] et que nous souhaitons intégrer à terme pour raffiner notre méthode de détermination du poids des noeuds.

5.2.4. Résultats

Les résultats sur un premier jeu de données de 3 arbres sont présentés dans le tableau 4. Les deux premières plantes, un pommier et un noyer, sont reconstruites à partir de mesures de digitalisation par des biologistes [CSKG03, SRG97]. Le troisième est un arbre simulé par L-systems [PL90].

Le codage différentiel représente le même arbre que l'arbre original, à la quantification près des coefficients de différence des points de contrôle sur 8 bits. Le modèle compressé sans les différences donne une représentation plus légère (entre 0.54 pour l'arbre généré par L-système, et 0.72 pour le pommier), pour une densité équivalente de branches. Ces résultats sont encourageants car les regroupements et la quantification sont faits de manière relativement naïve. La figure 3 montre un exemple de recontruction du noyer avec et sans les coefficients différentiels.

6. Conclusion et perspectives

Nous avons proposé un codage différentiel pour des plantes représentées par une arborescence de cylindres généralisés. Les premiers résultats sont encourageants. De nombreuses pistes restent néanmoins à explorer.

Dans ce premier travail, les regroupements de branches est fait de manière simple en considérant uniquement le nombre de points de contrôle. Il serait intéressant de prendre en compte la précision et la taille des données issues de la quantification faite sur les points de contrôle pour déterminer les groupes.

Une étude de sensibilité sur l'impact des différents paramètres définissant notre modèle de branches sur la représentation finale peut nous permettre de caractériser leurs importances et ainsi créer de nouveaux niveaux de détails. En effet, certaines valeurs de paramètres peuvent varier très faiblement et avoir un impact minime sur la la représentation.

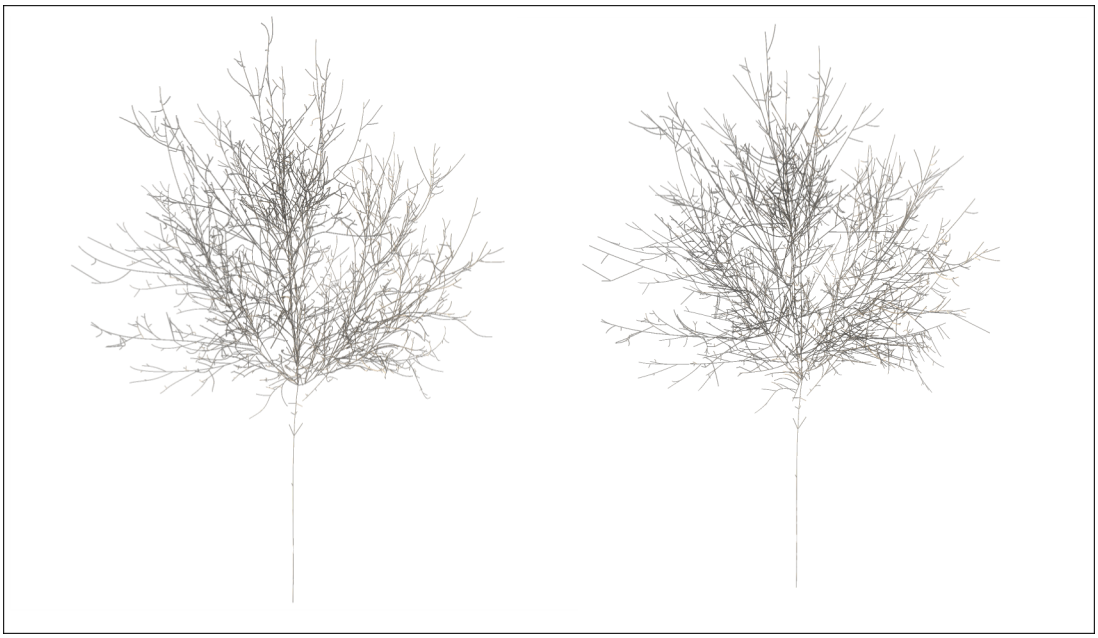


Figure 3: *Exemple de reconstruction d'une plante (Noyer numérisé [SRG97]) avec et sans les coefficients différentiels : à gauche compression sans pertes (74 % du modèle original), à droite, avec pertes (61 % du modèle original).*

Nom	Original	Compression *sans pertes* (ratio)	Sans les différences (ratio)
Pommier	109678	90668 (0.83)	78980 (0.72)
Noyer	580788	429047 (0.74)	353975 (0.61)
L-système	1519978	1028551 (0.68)	817687 (0.54)

Figure 4: *Résultats expérimentaux prennant en compte uniquement la géométrie (courbes de Bézier des branches) et les références nécessaires (références topologiques inter-branches et références aux branches moyennes). Valeurs données* ***en bits*** *(codage des réels : 32 bits, des différences : 8 bits).*

Il paraîtrait donc judicieux de différer la transmission de tels paramètres.

Pour des modèles assez réguliers, des valeurs prédictives, déduites de connaissances biologiques des plantes, pour certains paramètres tels que les angles de branchements peuvent être utilisées en première approximation sans nécessiter de transmission par le réseau.

Nous envisageons d'autre méthodes de compression des branches basées par exemple sur l'analyse en composante principale (ACP) des paramètres d'un groupe de branches permettraient de coder les différences sur une base de représentation des données hiérarchique et donc adaptée à notre problème.

Finalement, des tests sur différents types de réseaux et différents systèmes (Ordinateurs, PDA, etc.) nous permettront en pratique de mieux caractériser l'adaptation de notre méthode au contexte du streaming.

7. Remerciements

Ce travail a été financé par le projet ANR NatSim 05-MMSA-0004-01. Nous remercions C. Godin pour sa contribution au projet et sur les structures multi-échelles, E. Costes et H. Sinoquet pour nous fournir des bases de données de plantes digitalisées.

References

[Blo85] BLOOMENTHAL J. : Modeling the mighty maple. *ACM Computer Graphics (Siggraph'85) 19*, 3 (1985), 305–311.

[BMG06] BOUDON F., MEYER A., GODIN C. : *Survey on Computer Representations of Trees for Realistic and Efficient Rendering*. Tech. rep., LIRIS UMR 5205 CNRS, 2006.

[Bou04] BOUDON F. : *Représentation géométrique multi-échelles de l'architecture des plantes*. PhD thesis, Université de Montpellier II, 2004.

[COM*07] CHENG W., OOI W. T., MONDET S., GRIGORAS R., MORIN G. : An analytical model for progressive mesh streaming. In *MULTIMEDIA '07 : Proceedings of the 15th ACM international conference on Multimedia* (2007), ACM Press.

[CSKG03] COSTES E., SINOQUET H., KELNER J., GODIN C. : Exploring within-tree architectural development of two apple tree cultivars over 6 years. *Annals of Botany 91* (2003), 91–104.

[GO05] GU Y., OOI W. T. : Packetization of 3D Progressive Meshes for Streaming over Lossy Networks. In *Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN)* (2005).

[Hop96] HOPPE H. : Progressive meshes. In *SIGGRAPH'96 Conference Proceedings* (1996).

[KB04] KOBBELT L., BOTSCH M. : A survey of point-based techniques in computer graphics. *Computers and Graphics 28*, 6 (2004), 801–814.

[KLK04] KIM J., LEE S., KOBBELT L. : View-Dependent Streaming of Progressive Meshes. *Shape Modeling Applications* (2004).

[nat05] The Nature Simulation Project, 2005. ANR 05-MMSA-0004-01 `http://www.irit.fr/NatSim/`.

[PKL06] PARK S.-B., KIM C.-S., LEE S.-U. : Error resilient 3-D mesh compression. *IEEE Transactions on Multimedia 8*, 5 (October 2006), 885–895.

[PL90] PRUSINKIEWICZ P., LINDENMAYER A. : *The algorithmic beauty of plants*. Springer Verlag, 1990.

[PMKL01] PRUSINKIEWICZ P., MÜNDERMANN L., KARWOWSKI R., LANE B. : The use of positional information in the modeling of plants. *ACM Computer Graphics (Siggraph'01) 22*, 4 (2001), 289–300.

[RCB*02] REMOLAR I., CHOVER M., BELMONTE O., RIBELLES J., REBOLLO C. : Geometric simplification of foliage. In *Proc. of the Eurographics02 Short Presentations* (2002), pp. 397–404.

[SRG97] SINOQUET H., RIVET P., GODIN C. : Assessment of the three-dimensional architecture of walnut trees using digitising. *Silva Fennica 31*, 3 (1997), 265–273.

[Tau99] TAUBIN G. : 3D Geometry Compression and Progressive Transmission, 1999.

[YKK05] YAN Z., KUMAR S., KUO C.-C. : Mesh segmentation schemes for error resilient coding of 3-D graphic models. *Circuits and Systems for Video Technology, IEEE Transactions on 15*, 1 (2005), 138–144.

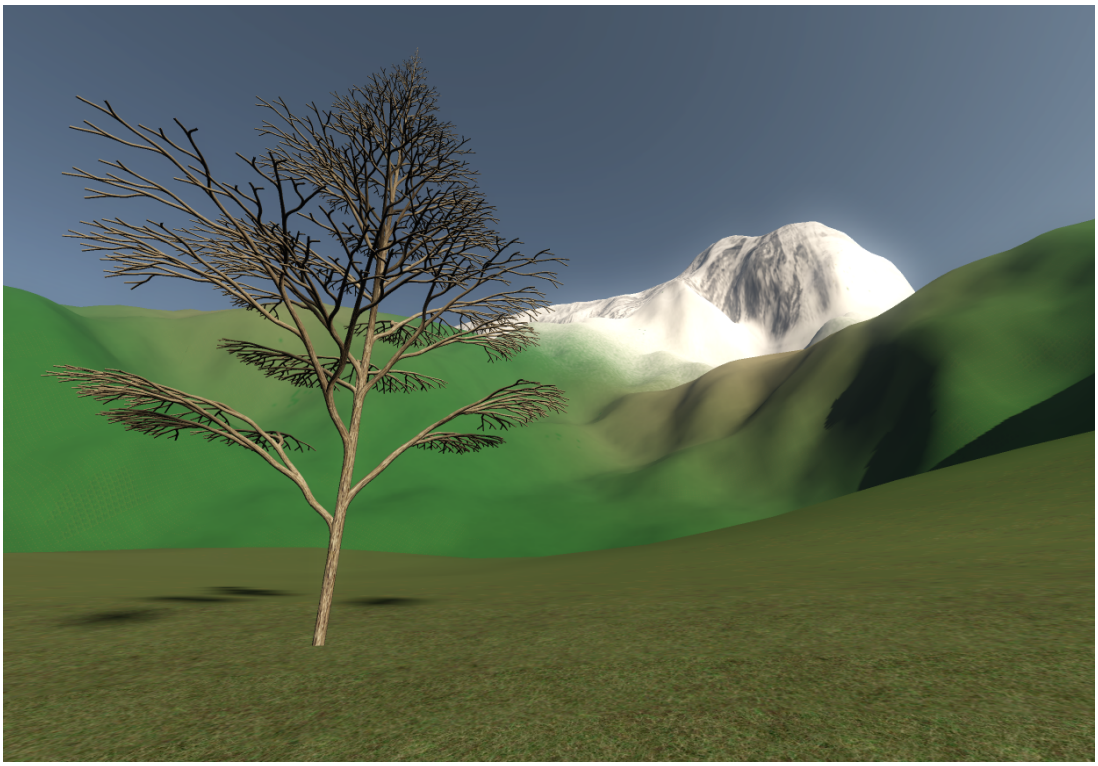


Figure 5: *Exemple d'utilisation de cylindres généralisés pour le rendu de plantes (ici un arbre généré par L-sytème).*

Développement d'un organe artificiel à partir d'une cellule unique

Sylvain Cussat-Blanc [1], Hervé Luga [1] et Yves Duthen [1]

[1]Université Sciences Sociales - TOULOUSE
2 rue du Doyen Gabriel Marty 31042 TOULOUSE CEDEX 9
{cussat,luga,duthen}@irit.fr

Abstract

Dans le but de créer des mondes virtuels peuplés de créatures artificielles toujours plus réalistes, différentes techniques de production de ces créatures existent. Certaines de ces techniques utilisent des blocs [K.S94, AJJ94, NHY07] ou des bâtonnets [MS99]. Dans cette approche morphologique du problème, ces structures peuvent être considérées comme les organes des créatures, c'est à dire des sous-parties de leur corps ayant des fonctions spécifiques. Une autre approche, souvent appelée embryogenèse artificielle, consiste à développer de petits organismes multicellulaires en partant d'une cellule unique [W.B03, AY07]. Dans cet article, nous proposons un modèle permettant d'unir ces deux approches. Dans un premier temps, il nous permettra de simuler le développement d'un organe en partant d'une cellule unique. Une fois une librairie d'organes créée, il nous suffira de les assembler pour obtenir une créature complète. Ce modèle est une forte simplification du processus de développement naturel. Il simule la spécialisation cellulaire, les réactions chimiques (d'une façon simplifiée) ainsi que la diffusion de substrats dans l'environnement.

Keywords : artificial embryogeny, cell specialization, artificial creatures, gene regulatory network, artificial environment, genetic algorithm.

1. Introduction

This paper shows the latest results of our model [SBHY07] of artificial embryogeny. In nature, cells are able to specialize their functions. This specialization allows the creation of cells groups that perform the same function, commonly named tissues. To perform a specific action, tissues are grouped in organs. The cell specialization happens at different stages in the organism's development. The internals of this specialization is coded in the DNA molecule as a gene regulatory network and regulated during the growth with different proteins contained in the foetus neighborhood. In our project, we want to simulate this specialization using different levels of detail. This paper presents a model that forgets the complexity of chemistry and physics to focus on the complexity of cell's mechanisms.

This paper is organized in three sections. Section 1 presents the related works about artificial creatures development, presenting artificial morphogenesis, cellular automata and already existing works about cell specialization. Section 2 presents our model of cellular development, starting with a description of the environment's operating and the mechanisms used by our artificial cell to interact with the environment. Section 3 presents first experiments using this model. The possibilities of the model are proved by the building of two organisms with different functions. One is able to harvest and eliminate substrate and another one move a specific substrate.

2. Related works

2.1. Artificial creatures

Several works tries to generate artificial creatures. For example, Karl Sims [K.S94] uses blocks with different properties as size, shape, contact sensor positions or blocks layout. Komosinski also creates Framsticks creatures [MS99] using an equivalent architecture: blocks are replaced by sticks but the creature functioning is comparable to Karl Sims' work: he uses a neural network to coordinate the creature moves.

Sims' work was improved by Nicolas Lassabe by using a more complex environment [NHY07]. Lassabe's creatures are able to climb a stairway or to practice skateboard.

The aforementioned creatures use high level components to create their morphology and their behavioral controller. A more biological-inspired approach was introduced by Dawkins in [R.D86]. Using simple rules to draw continuous segments, he developed a software able to create graphic creatures. The addition of behaviors in these simple life forms allows the creation of a complex 2-D virtual world [J.V98b, J.V98a] where small filiform creatures co-evolve in an environment composed of energy sources. Each creature has a vital energy level and must survive in the environment, looking for food. EvolGL [CMF04] is a 3D pond life project where creatures have different classes, like herbivorous, carnivorous or omnivorous, which allows the emergence of survival strategies.

Using lower level components, cellular automata use neighborhood rules to evolve a cell matrix. The rules give the *t+1* state of each cell according to the cell neighbor's *t* state. Using this method, John H. Conway [M.G70] creates interesting patterns like gliders, pulsars, spaceships, etc.

To make a bridge between their methods, we can express the following hypothesis: blocks and sticks can be considered as organs, that is to say body parts of the creature able to carry one or more specific functions. Using developmental techniques of creature growing, we could create this organs starting from a single cell. In this way, the cell must be able to specialize itself into a cell more adapted to the environment. The cells organization in tissues (it means in cell groups that have the same function) and then the tissues organization will allow the creation of organs. After creating a library of organs, we will just have to assemble them to create an creature adapted to the environment.

2.2. Cell Specialization

Different works about cell specialization already exist. They all use a Genetic Regulatory Network (GRN), just like in the nature.

In nature, the cells of an organism can have different functions, all of which are specified in the organism's genome and regulated by a Gene Regulatory Network (GRN) [E.H06]. Cells get input signals from the environment thanks to receptor proteins. The GRN, described in the organism's genome, use these signals to activate or inhibit the transcription of different genes in the messenger RNA, the future cell's DNA protein template. The expression of those genes will specify the cell's functions. Figure 1 shows (in a very simplified way) the functioning of the GRN.

This natural model has been offspring by Banzhaf in [W.B03]. In this work, each gene beginning is marked by a starting pattern, named "promoter". Before the coding of the gene itself, enhancer and inhibitor sites allow the regulation of its behavior. In [AY07], Chavoya and Duthen introduced another model in which the gene regulation system is encoded in the beginning of the genome. It consists in a series of inhibitor sites, enhancer sites and regulatory proteins. The production of each regulatory protein is conditioned by the inhibitor/enhancer sites. The concentration of this protein determines the cell function's activation or the inhibition : if the concentration level is over a threshold, the gene is activated and so are the corresponding functions.

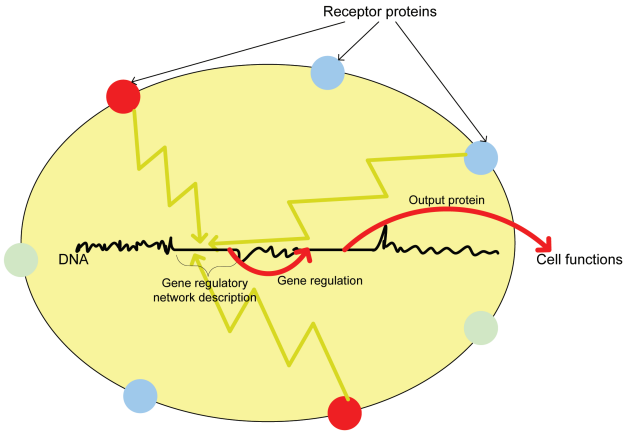


Figure 1: *Scheme of the GRN action in the cell duplication.*

A different approach is the Random Boolean Network (RBN) first presented by Kauffman [S.K] and reused by Dellaert [FR94]. A RBN is a network where each node has a boolean state: activate or inactivate. The nodes are interconnected by boolean functions, represented by edges in the net. The state of a node at time $t + 1$ depends on its particular boolean function applied to the values of its inputs at time t. The mapping to the gene regulatory network is simple: each node of the net corresponds to a gene and each boolean function represents the activity regulation of the gene. The cell function will be determined during the interpretation of the genome.

Our model of cell specialization is inspired by Dellaert's model but it is quite different. We use efficiency and transfer coefficients to allow the specialization of our cell. Let us detail our cell development model beginning with the environment, the cell mechanisms and especially the GRN.

3. Our cellular developmental model

3.1. The environment

To reduce the simulation computation time, we implement the environment as a 2-D toric grid. This choice allows an important decrease in the simulation's complexity. The extension of this model to a 3-D environment is as easy as us-

ing a matrix instead of a grid and adding the top and bottom directions to the diffusion system.

The environment contains different substrates. They diffuse in the grid, minimizing the variation of substrate quantities between two neighbor crosses of the grid. This diffusion acted in two steps, as illustrated by figure 2

- First, the substrate diffuses to the 4 cardinal points.
- Then, if the substrate quantity is sufficient, the substrate diffuses to the diagonal crosses.

A substrate has different properties like diffusion speed or color, and can interact with other substrates. This interaction between substrates can be viewed as a chemical reaction: using different substrates, the transformation will create new substrates, emitting or consuming energy. For example, the transformation $2A + B \rightarrow C\ (+50)$ denotes that, using 2 units of substrate A and 1 unit of B, a unit of C is created, emitting 50 units of energy. From a biological point of view, C can be seen as a waste from a cell which has the ability to convert A and B in energy.

The environment is now defined. To modify this environment, we add cells with different abilities and specific goals. The next section details their functioning.

3.2. The cells

Cells are represented in the environment as squares on the grid. Each contains sensors and has different abilities (or actions). A classifier system [JJ78] allows the cell to select the best action to perform at any moment of the simulation. Finally, a representation of a GRN is inside the cell to allow specialization embedded over duplication.

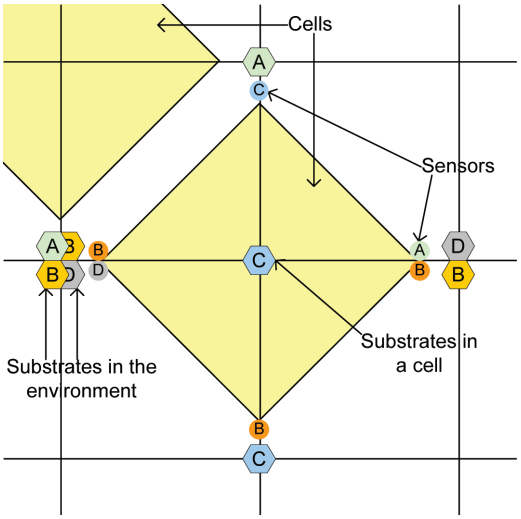


Figure 3: *Scheme of a cell in an artificial environment. It contains substrates (hexagons) and corresponding sensors (circles)*

3.2.1. Sensors

Each corner of the cell square contains different density sensors that allow the cell to measure the amounts of substrates available in the cell's Von Neumann neighborhood. For each substrate in the environment, a corresponding sensor exists. Only this corresponding sensor can compute the density of the substrate. The list of available sensors and their position in the cell is described in the genetic code.

For example, on figure 3, the cell has sensors for B and D substrates on the left corner. The results of the measure of the corresponding substrate densities are :

- 2 units for B substrate because of the presence of 2 units of B substrates on the left cross of the cell,
- 1 unit for D substrate.

3.2.2. Actions

To interact with the environment, cells can perform different actions:

- The *substrate transformation* allows the cell to trigger a substrate reaction as described in Section 3.1. To start, all the needed substrates on the left part of the equation must be present in the cell, that is, the needed substrates must be in the same intersection as the cell. In result of the reaction, the vital energy is increased or decreased (depending of the reaction properties), the needed substrates are destroyed and the new substrate is created.
- The cell can *absorb* or *reject* substrates in the environment. These two actions allow the cell to move substrates from a place to another. These actions, and particulary the first one, are important to trigger a substrate transformation.
- The *duplication* action allows the cell to create a new cell. We give details about this action in next section.
- *Move* allows the cell to go to a free intersection (two cells cannot be at the same place) at distance 1 of the current location.
- *Survive* is an action that allows the cell to wait for a signal of the environment to do something.

The previous list is not final. Our model must be able to allow us to add new actions easily. Like sensors, all actions are not available for the cell: the genetic code will give the available action list.

Using the information given by the sensors, the classifier system will allow the cell to select the best action to perform. The classifier system is composed of a list of rules. Each rule contains three parts:

- The precondition describes when the action can be triggered. It is composed of a list of sensor value intervals that describe the best substrate densities in the neighborhood to trigger the action.
- The action part gives the action that must be performed if the corresponding precondition is respected.

- The action priority that selects only one action if more than one can be performed.

In the list of possible actions, the cell can duplicate itself. We are now going to see this action in detail.

3.2.3. Duplication

The duplication is an action that can be selected by the classifier system. It can be performed if the "use" conditions are satisfied:

- The cell must have at least one free neighbor cross to create the new cell.
- The cell must have a enough vital energy to perform the duplication.
- The environmental specifications must be respected. Their condition are described during the environment modelling as a list of substrates needed.

The new cell created after the duplication is completely independent. It has its own classifier system, its own action list, etc. During the duplication, the cell can be specialized to optimize a group of actions instead of others actions. In nature, this optimization is done by the GRN. In our model, we imagine a simplification of the GRN. Each action has an efficiency coefficient that corresponds to the action optimization level : the higher the coefficient, the lower the cost of vital energy. Moreover, if the coefficient is null, the action is not yet available for the cell. Finally, the efficiency coefficient sum does not vary during the simulation. In other words, if an action is optimized increasing its efficiency coefficient during a duplication, another efficiency coefficient (or a group of them) has to be decreased.

The cell is specialized by varying the efficiency coefficients during the duplication. The rules of their variations are given by the GRN. We use a network to simulate the GRN:

- the network's nodes represent cell actions with their efficiency coefficients,
- the network's edges are weighted. The edge's weight (a real in the interval [0,1] represents the efficiency coefficient quantity that will be transferred during the duplication.

Figure 4 is an example of our GRN. (A, 35%), (B, 25%), (C, 17%), (D, 23%) are cell actions with their associated efficiency coefficient. The edge between 2 actions represents the amount of efficiency coefficient that will be transferred during duplication. For example, the weighted edge between A and B means that after one duplication, 30 percents of the A action efficiency coefficient will be transferred to the B action. After four duplications, we can see that the actions B and C respectively passing from has been optimized at the detriment of the actions A and D. According to this simple example, we can say that the cell function of the organism has been specialized during the duplication.

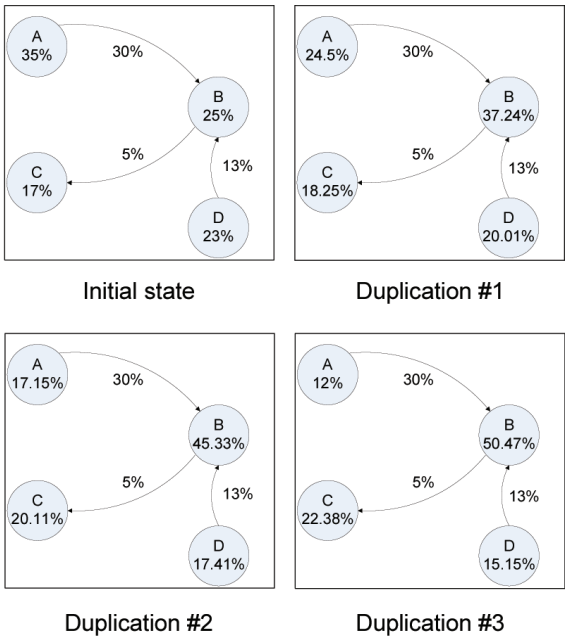


Figure 4: *Modelization of an example of the Gene Regulatory Network. A, B, C and D are 4 actions with their efficiency coefficient. The transfer coefficients are given by the arrows.*

4. First results

We have implemented this model in Java using a multi-threaded architecture: each cell functioning is simulated by a thread. We will now decribe our first experiments.

4.1. Harvesting substrate

The first experimentation consists in an environment substrate harvesting. The environment is composed of three different substrates: a red, a blue and a pink. The aim of the organism is to consume the maximum amount of red substrate. To that end, the cell can:

- duplicate itself (needs one blue substrate unit and vital energy),
- absorb or reject any substrate (consume vital energy),
- transform one red substrate unit into one blue and one pink (produce vital energy).

We can observe that the red substrate is the fuel of the cell, the blue substrate can be considered as a duplication material and the pink as a waste.

We dispatch 1900 red substrate units in the environment and we put a single cell in the center of the environment. We use a genetic algorithm to evolve the genetic code of the cell. After computation, the best organism harvest more than 1700 units of the red substrate. A strategy emerges during

the try: the cell uses the duplication to move in the environment and the organism deploys itself on a complete column moving from the left to the right of the environment[†] (figure 5). During the genetic algorithm convergence, we observe that the organism begin to learn to move, using the duplication and doing it on only 4 or 5 crosses. After many generations, the organism is able to harvest the totality of substrates available on an environment diagonal just using the previous move manner. Later, the organism finds a solution to harvest the remaining substrates in the environment. It uses a duplication of its pattern into an orthogonal way of its move to finally cover the totality of a column.

4.2. Developing a transfer system

The next experimentation consists in developing a transfer system, which means a structure able to transport substrate from a point to another. To do that, we imagine an environment composed of 2 substrates:

- A red that the organism must move. This substrate has the specificity not to diffuse in the environment, in order not to impact on the organism work.
- A gray that will be used by the cell as fuel and duplication material.

The cell can perform the following actions:

- duplicate (needs one gray substrate and vital energy),
- absorb or reject substrate (consume vital energy),
- transform one gray substrate in vital energy.

We squish 10 red substrate units into a specific cross of the grid (in the top left of the environment) and diffuse gray substrate all over the environment. The creature's score is given by the squared sum of the red substrate distance to the goal point (in the bottom right of the environment). The parameters of the genetic algorithm are:

- mutation rate: 5%,
- crossover rate: 65%,
- selection algorithm: 7 tournament competition with elitism,
- substitution algorithm: worst individuals,
- population size: 500 individuals,

Figure 6 shows the convergence curve of the genetic algorithm. It shows the variation of the minimum, the average and the maximum fitness of the population for each generation. The genetic algorithm's aim is to maximize the fitness, which is the creature score. A relevant organism appears quickly. After 3 generations, the organism is able to move the red substrates but not in the right direction. After 10 generations, it is able to move closer to the goal point. The genetic algorithm converges after 22 generations (the average fitness is close to the best).

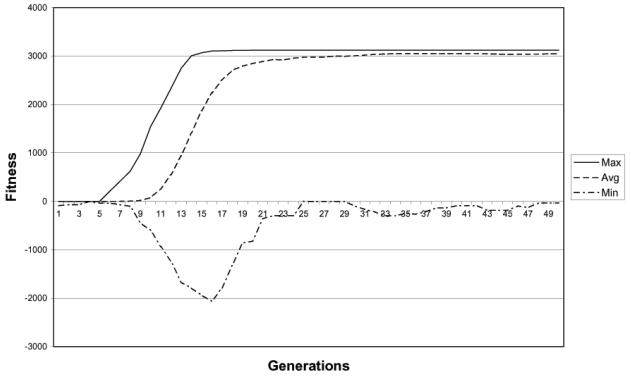


Figure 6: *Smooth curve of the minimum, average and maximum organism fitness. The genetic algorithm must minimize the sum of the squared distance from the red substrate to the goal point.*

Figure 7 shows the development of the best organism[‡]. We can see that only the cell on the way from the initial point to the end point are created. Moreover, the organism uses absorption and rejection actions to transfer the substrate gradually. Cells that overtake the final point die quickly for not interacting in the substrate transfer. During the convergence of the genetic algorithm, it is interesting to observe the evolution of the organism strategy towards the best solution. The first step is to learn to survive in the environment, absorbing gray substrate and transforming it in vital energy. The next step is to learn to duplicate in the right direction. Intermediate solution organisms are able to transport the red substrate form the initial point near to the goal. The organism also develop itself in all the environment, scattering some units of the substrate in the environment. As shown in figure 7, this organism deploy itself just only on the best trajectory, decreasing the substrate scattering probability.

5. Conclusion and future works

We propose a model of cellular development. This model is based on a strong simplification of the natural development. We forget the physics rules and the atomic and molecular interactions to lean over the cell abilities. Using a genetic algorithm and specific environment, we create different organs with different functions.

In order to build more, the next step can be an organ able to harvest different substrates and transform them into vital energy and depose wastes on a specific position. When we have a sufficient organ library, we want to associate different organs to develop an organism. It will be able to develop

[†] A video of this creature is available on the website http://www.irit.fr/∼Sylvain.Cussat-Blanc

[‡] A video of this creature is available on the website http://www.irit.fr/∼Sylvain.Cussat-Blanc

itself in the environment using the available resources. The final step of this project is to put many different creatures in the same environment. Using co-evolution, we hope to see the emergence of an organization of the creatures, some of then could use waste from others to create their own energy. After an undefined evolution time, a food chain could appear in the substrate transformations, creating a viable ecosystem.

An interesting possible improvement to this model could be a complexification of the GRN. For the moment, it just use transfer coefficient that are defined at the beginning of the simulation (in the genetic code of the first cell). It could be interesting to use the sensors and then the neighborhood of the cell to modify their transfer coefficient.

References

[AJJ94] A.FUKUNAGA, J.MARKS, J.T.NGO: Automatic control of physically realistic animated figures using evolutionary programming. *Proc. of Third Annual Conference on Evolutionary Programming* (1994), 76–83.

[AY07] A.CHAVOYA, Y.DUTHEN: Evolving an artificial regulatory network for 2d cell patterning. *Artificial Life* (2007).

[CMF04] CARBAJAL S., M.B.MORAN, F.G.MARTINEZ: Evolgl: Life in a pond. *Artificial Life XI* (2004), 75–80.

[E.H06] E.H.DAVIDSON: The regulatory genome: gene regulatory networks in development and evolution. *Academic Press* (2006).

[FR94] F.DELLAERT, R.D.BEER: Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV* (1994), 246–257.

[JJ78] J.H.HOLLAND, J.S.REITMAN: Cognitive systems based on adaptive algorithms. *Pattern-Directed Inference Systems* (1978).

[J.V98a] J.VENTRELLA: Attractiveness vs efficiency (how mate preference affects location in the evolution of artificial swimming organisms). *Artificial Life VI* (1998), 178–186.

[J.V98b] J.VENTRELLA: Designing emergence in animated artificial life worlds. *Internationnal Conference on Virtual Worlds* (1998), 143–155.

[K.S94] K.SIMS: Evolving 3d morphology and behavior by competition. *Artificial Life IV* (1994), 28–39.

[M.G70] M.GARDNER: The fantastic combinations of john conway's new solitaire game life. *Scientific American 223* (1970), 120–123.

[MS99] M.KOMOSINSKI, S.ULATOWSKI: Towards a simulation of a nature-like world creatures and evolution. *European Conference on Artificial Life* (1999).

[NHY07] N.LASSABE, H.LUGA, Y.DUTHEN: A new step for artificial creatures. *Artificial Life* (2007).

[R.D86] R.DAWKINS: The blind watchmaker. *Longman Scientific & Technical* (1986).

[SBHY07] S.CUSSAT-BLANC, H.LUGA, Y.DUTHEN: A developmental model to simulate natural evolution. *International Conference on Computer Graphics and Artificial Intelligence 3IA* (2007).

[S.K] S.KAUFFMAN: Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theorical Biology 22*, 437–467.

[W.B03] W.BANZHAF: Artificial regulatory networks and genetic programming. *Genetic Programming Theory and Practice* (2003), 43–62.

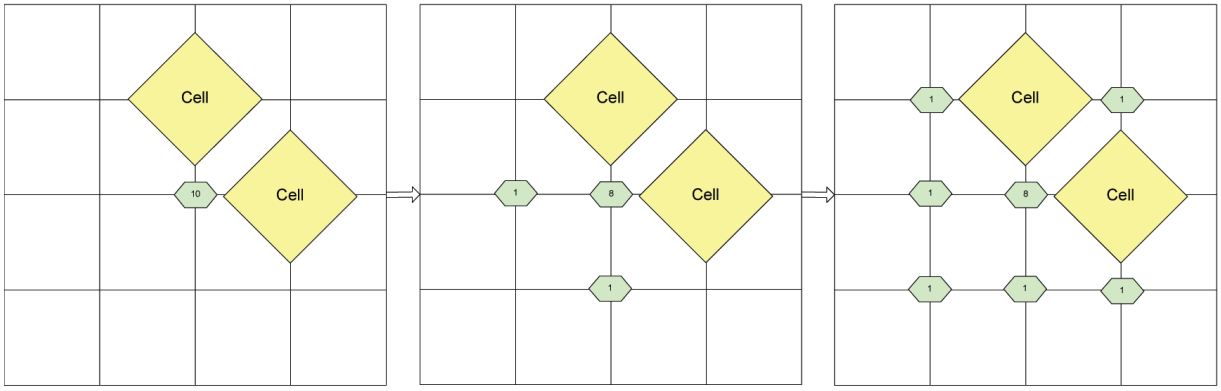


Figure 2: *Example of diffusion of a substrate in the environment.*

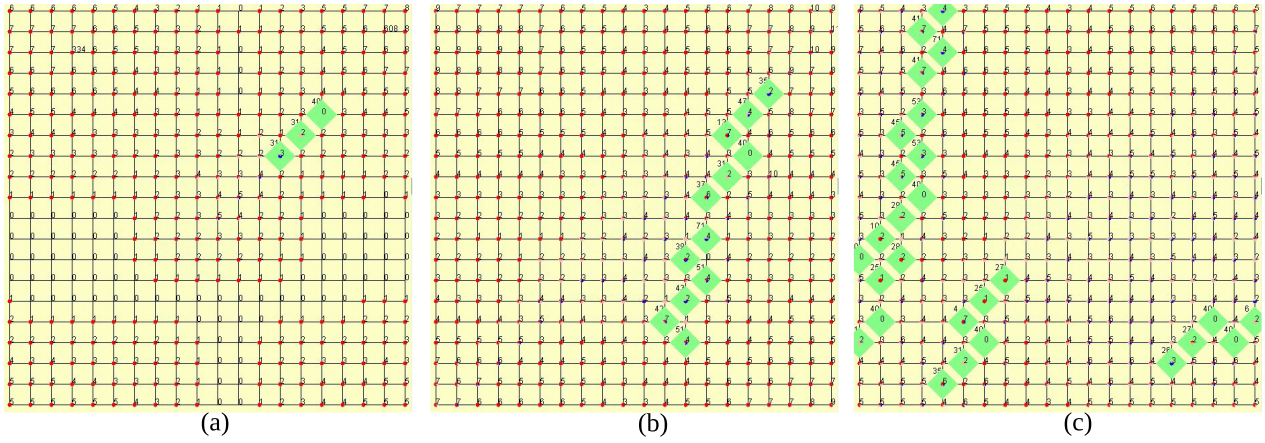


Figure 5: *A creature able to harvest a specific substrate. (a) Beginning of the development, the creature moves using duplication. (b) The creature becomes larger. (c) The creature harvests an entire column and moves from the left to the right of the environment.*

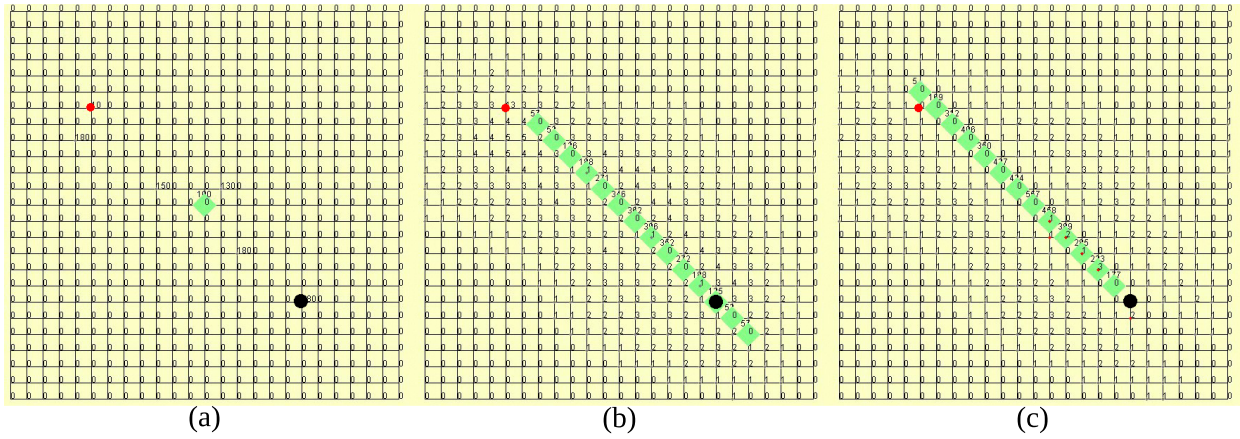
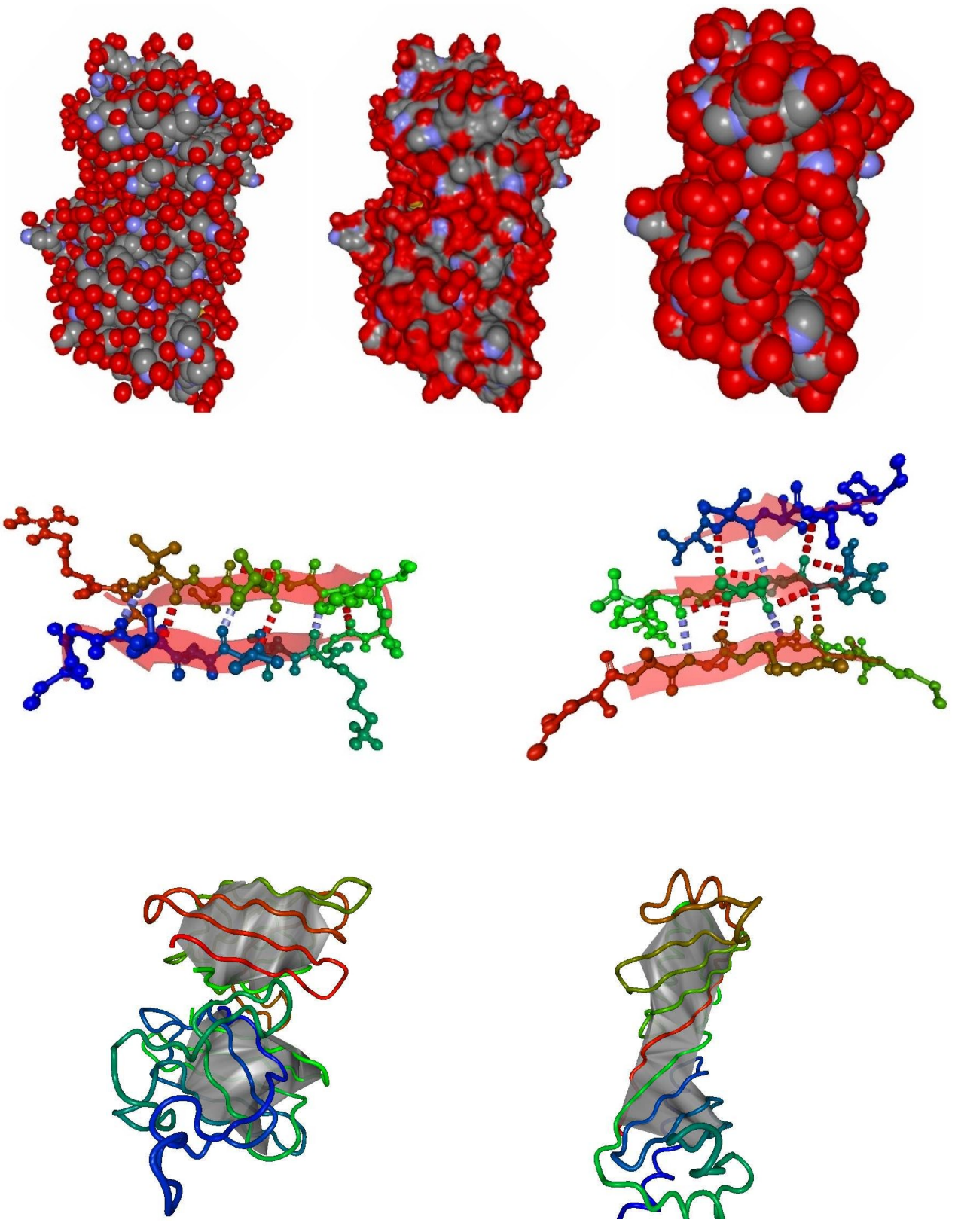
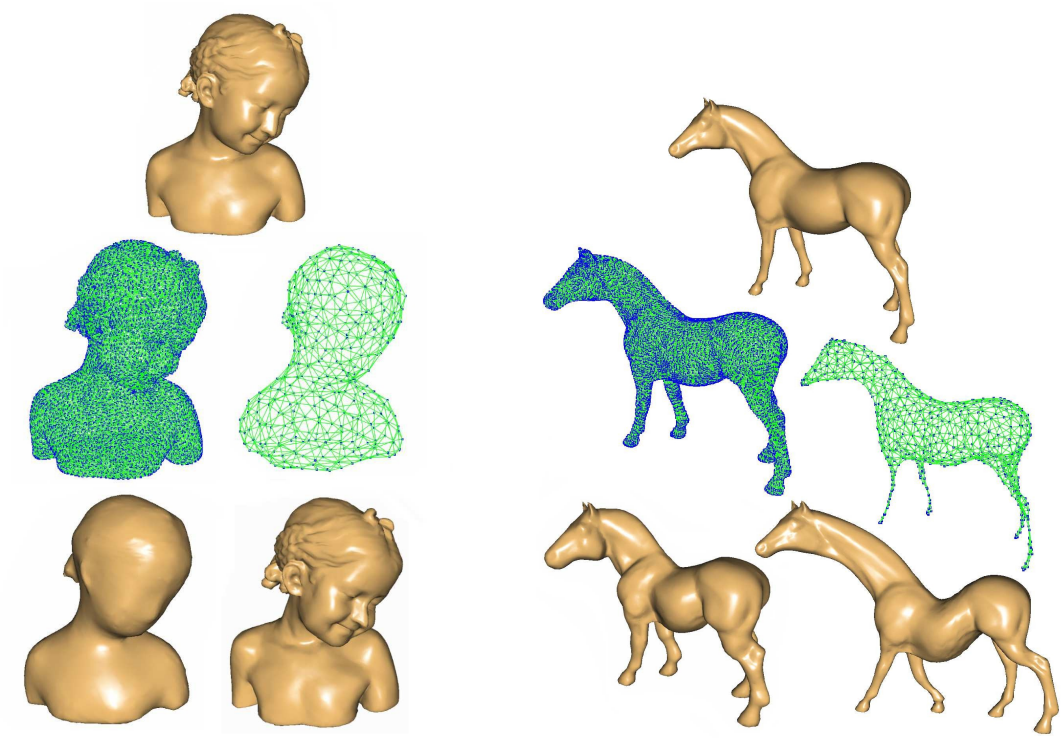


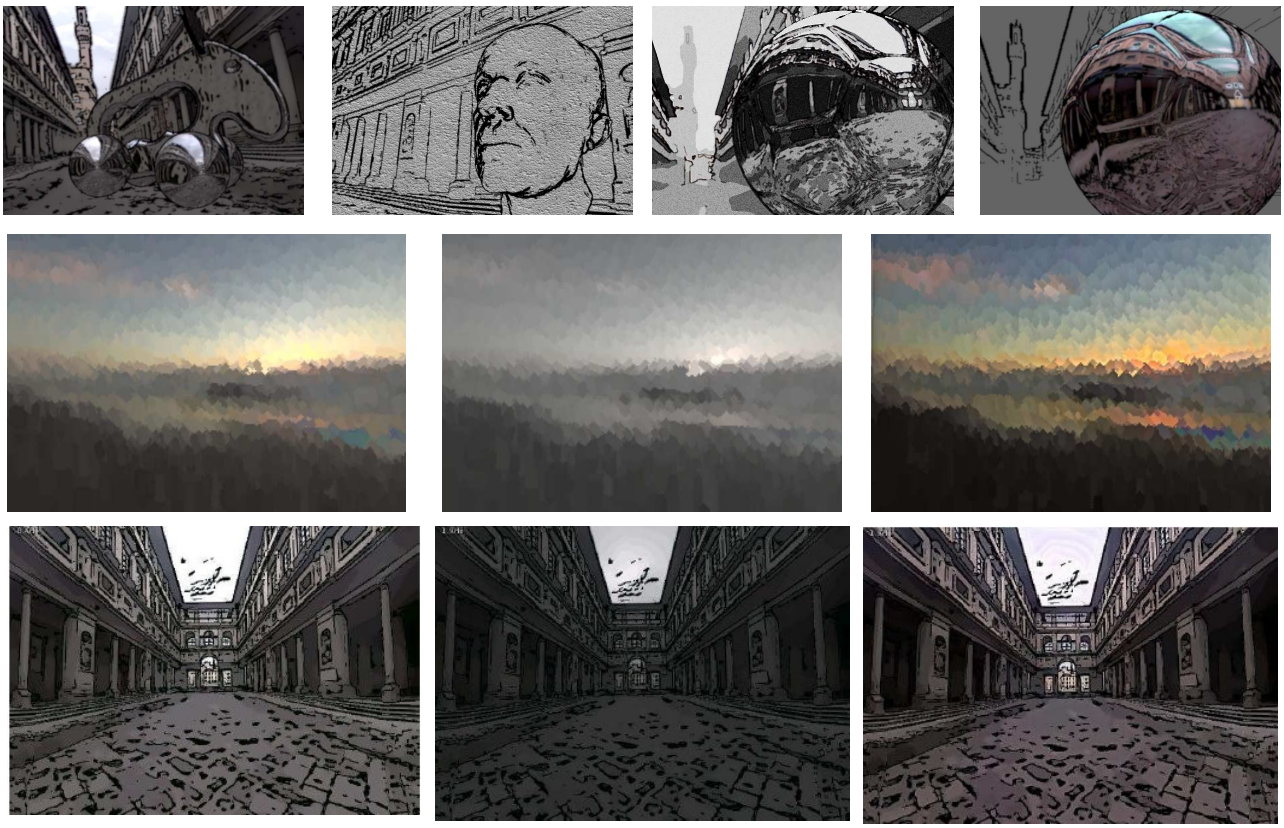
Figure 7: *Our artificial transfer system. (a) Beginning of the simulation. (b) The creature develops itself to create the structure. (c) The creature transfers the substrate from the initial state (top left) to the final state (bottom down).*



Simulation de la Surface de Feuillets Beta en Modélisation Moléculaire,
Loïc Nolin, Aassif Benassarou, Manuel Dauchez, Yannick Rémion ,



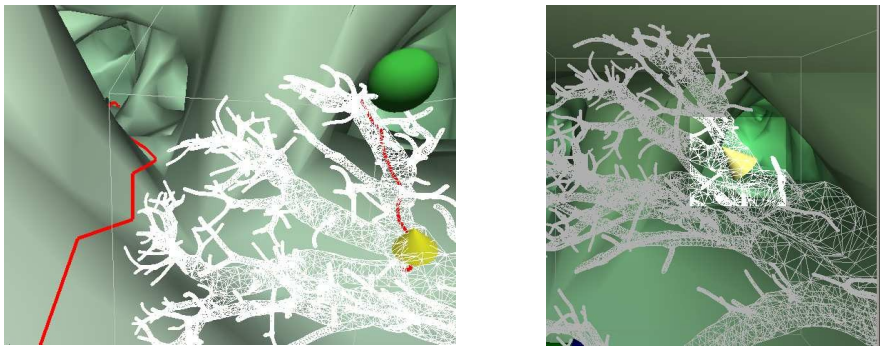
Un modèle générique pour la manipulation de maillages multirésolution,
Pierre Kraemer, David Cazier, Dominique Bechmann,



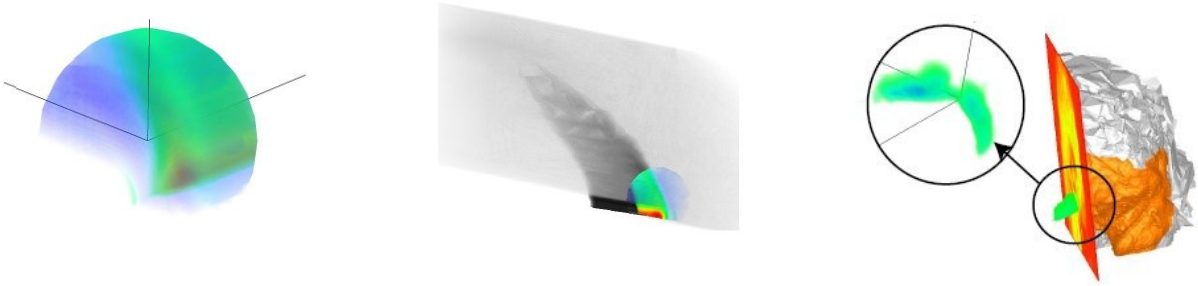
tylisation d'objets éclairés par des cartes d'environnement HDR,
Romain Vergne, Xavier Granier,

Chaîne de Traitement pour la Numérisation et le Rendu Réaliste de Peintures d'Art,
Frédéric Larue, Lucas Ammann, Jean-Michel Dischler,

Effet de flou de profondeur pour la navigation en environnement virtuel avec vue à la première personne,
Sébastien Hillaire, Anatole Lécuyer, Rémi Cozot, Géry Casiez,

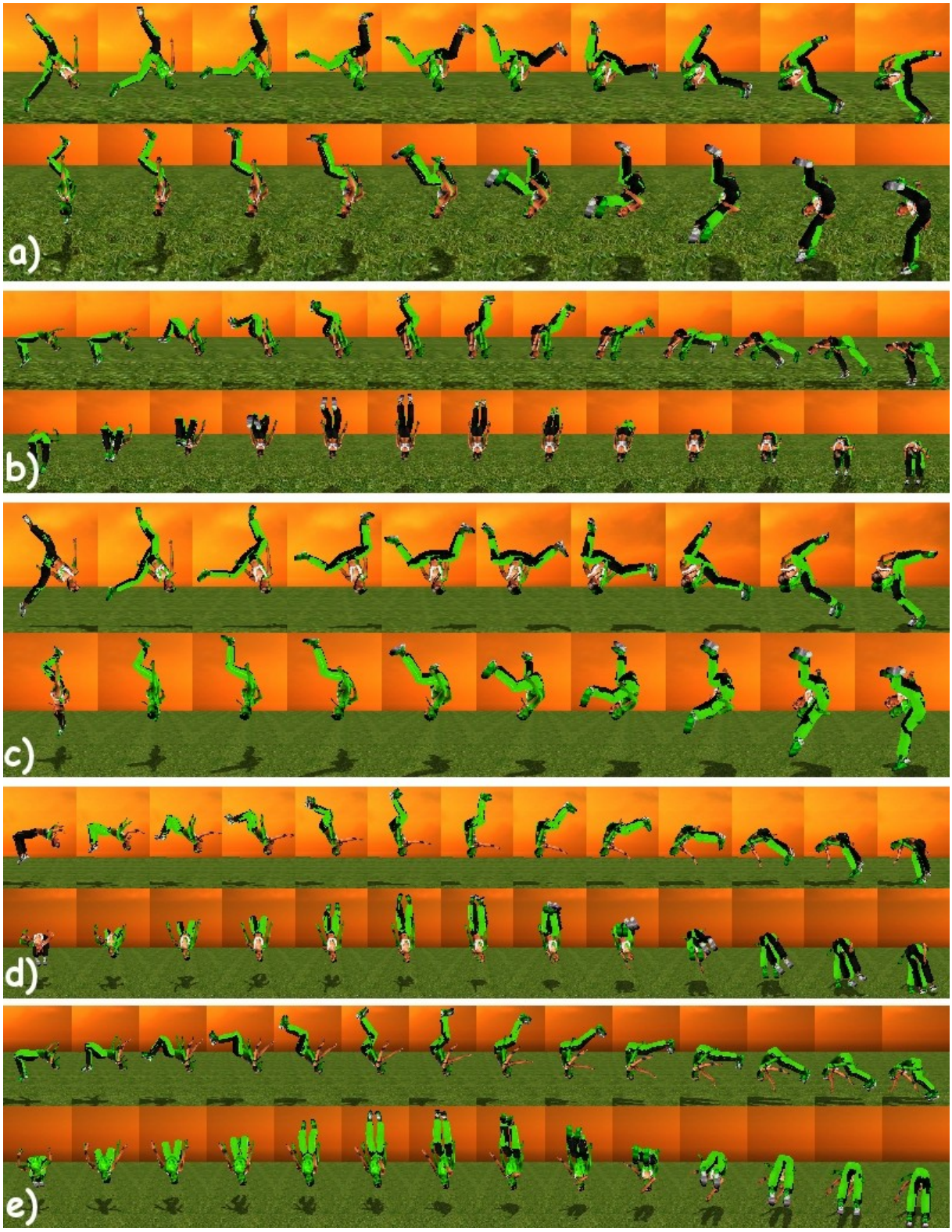


Navigation 3D virtuelle : positionnement et technique de sélection dans un affichage volumétrique complexe,
Thomas JUND, David CAZIER, Dominique BECHMANN,

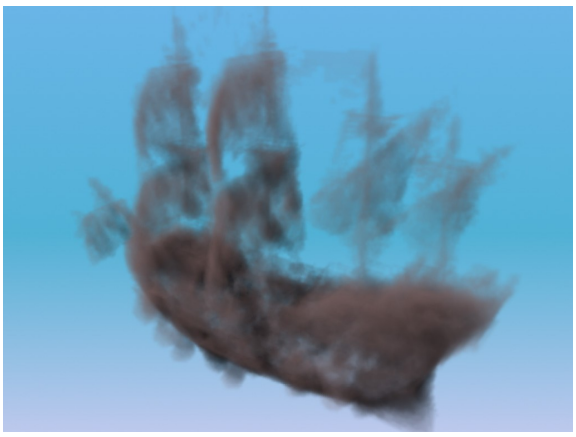


Visualisation Focus+Contexte pour l'Exploration Interactive de Maillages Tétraédriques
Sébastien Barbier et Georges-Pierre Bonneau

Insertion de détail dans des figures autosimilaires
Houssam Hnaidi, Eric Guérin et Samir Akkouche

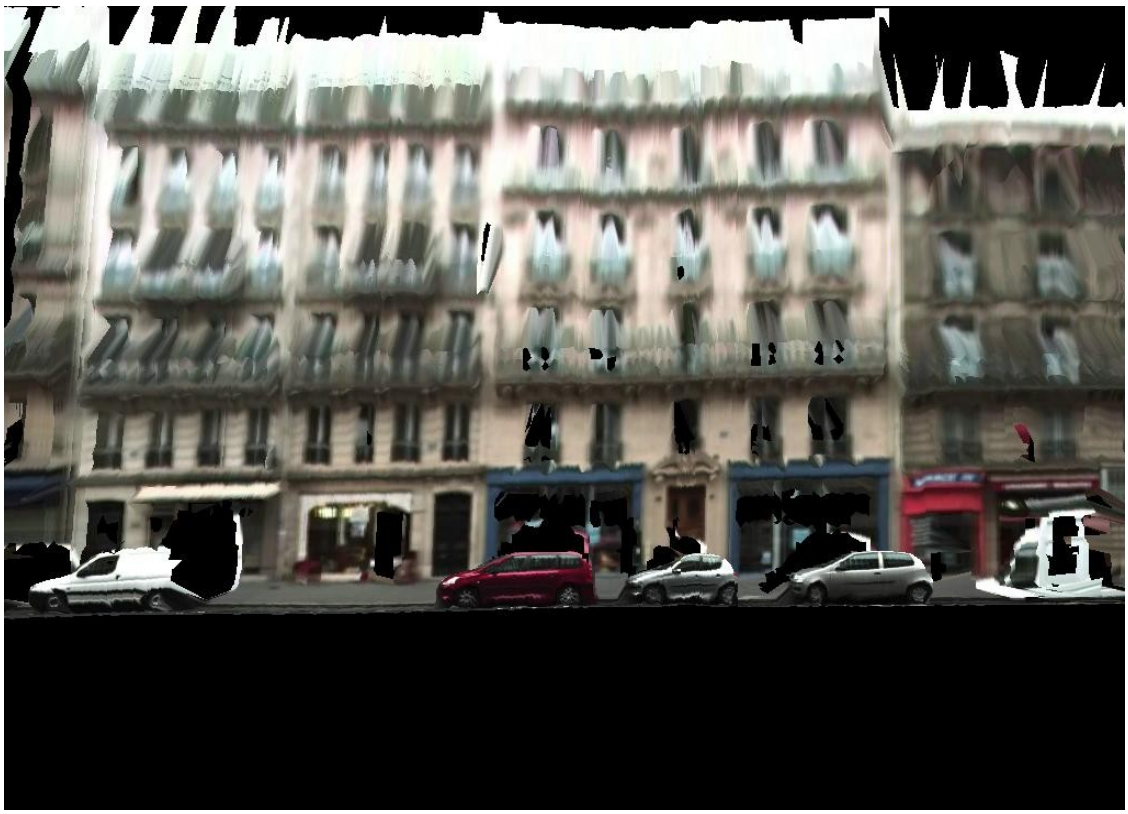


Adaptation dynamique de mouvements aériens,
Ludovic Hoyet, Richard Kulpa, Taku Komura, Franck Multon,

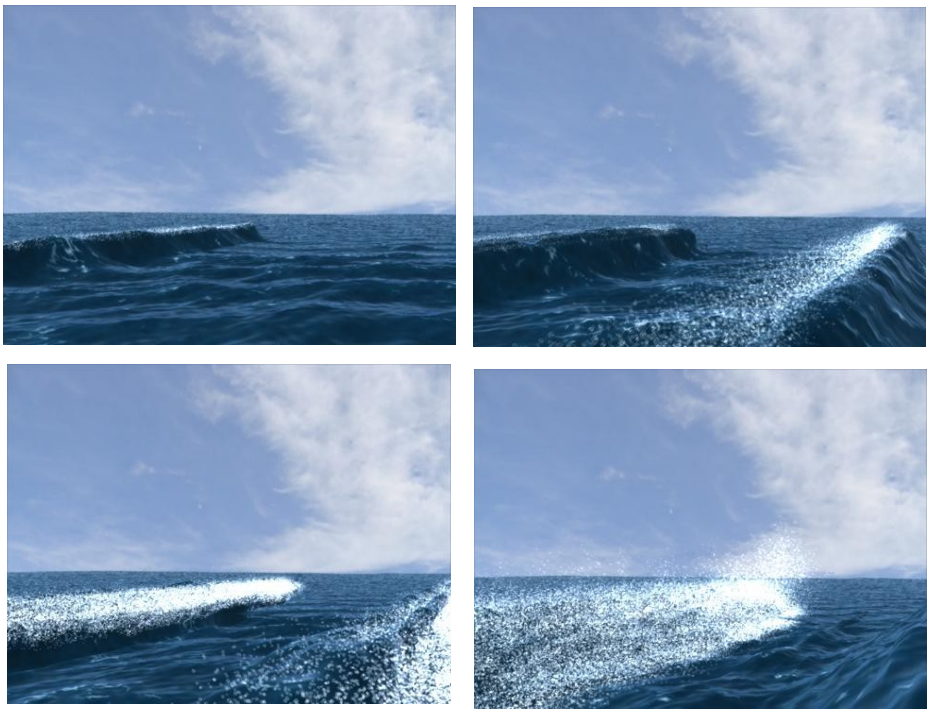


Modélisation de formes vapeureuses à l'aide d'IFS,
Dmitry Sokolov, Christian Gentil, Marc Neveu,

Affichage non aliasé pour l'illumination globale par voxels,
Lukasz Piwowar, Malgouyres Rémy,

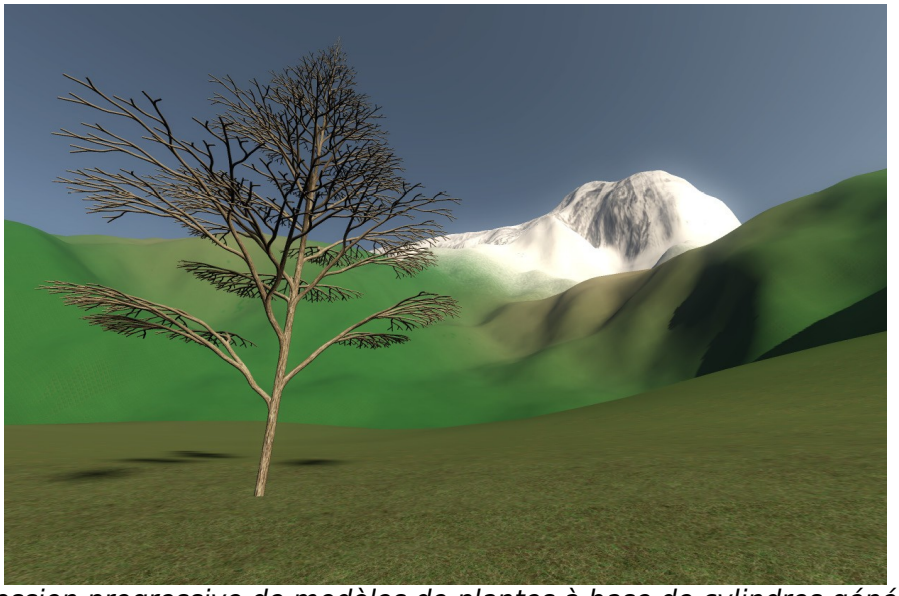


Texturation d'environnements urbains par système mobile avec un couplage Caméra/Télémètre Laser,
Jean-Emmanuel Deschaud, Xavier Brun, François Goulette



Une approche multirésolution lagrangienne pour la simulation de vagues déferlantes,
Emmanuelle Darles, Benoît Crespin, Djamchid Ghazanfarpour,

Rendu réaliste de nuages temps-réel,
Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, Cyril Crassin,



Compression progressive de modèles de plantes à base de cylindres généralisés
S. Mondet, F. Boudon, J.C. Hoelt, G. Morin, R. Grigoras, C. Pradal, M. Paulin

AFIG 2002
AFIG 2005