Rapport de stage de DEA Algorithmique

Maillage de surfaces à partir de points digitalisés, sans structure interne.

Sylvain Pion *†

Mars-Juin 1996

Motivations :

De nombreux secteurs industriels font appel à la CFAO [1], principalement l'industrie automobile. La société Matra Data Vision développe depuis de nombreuses années des logiciels dans ce domaine.

Un des aspects de la CFAO est ce que l'on appelle le "reverse engineering". Il consiste à modéliser une maquette ou un prototype physique, afin de pouvoir ensuite le modifier sur ordinateur, avant de donner des ordres à des machines outils pour qu'elles produisent l'objet en série, ou bien même un exemplaire unique en stéréo-lythographie, par exemple.

Pour ce faire, on dispose d'instruments de digitalisation qui fournissent les coordonnées de points dans l'espace, situés à la surface de l'objet.

Le but est de former, à partir de ces points, un maillage (une triangulation surfacique) décrivant au mieux la surface. Ce rapport s'attache au cas où les points ne sont pas structurés, et se limite à un maillage passant par les points donnés.

* ☺-mail : Sylvain.Pion@ens.fr

† Directeurs de stage : Nazih Saadé (n-saade@paris1.matra-dtv.fr), Jean-Daniel Boissonnat (Jean-Daniel.Boissonnat@sophia.inria.fr)

1. Conception et fabrication assistées par ordinateur.

Table des matières

1 Définition du problème

1.1 Le sujet

Certains capteurs (ou palpeurs) fournissent des ensembles de points ordonnés, soit en grilles, soit en lignes. C'est le cas le plus courant. Un algorithme de maillage peut utiliser à son avantage de telles structures.

Par exemple, Jean-Daniel Boissonnat et son équipe maillent des ensembles de points "alignés" obtenus par des coupes parallèles. Le cas des grilles est un peu plus simple, et a fait l'objet de nombreux articles et thèses (voir la bibliographie à la fin de ce rapport).

Parfois, les données sont de plusieurs types, des lignes définissent les arêtes, et des grilles les larges plages à faible courbure. De plus, pour définir un objet suffisamment, on a souvent de nombreuses grilles, et il faut alors traiter ce cas différemment du cas d'une grille unique.

Il y a deux méthodes possibles :

- soit on maille les diverses parties structurées de l'entrée, puis on fusionne ces maillages,
- soit on crée un maillage en fusionnant directement les entrées.

La première méthode doit s'adapter à tous les cas possibles en pratique, ce dont on ne peut a priori pas faire de liste exhaustive. Alors que la deuxième permet de s'adapter à tous les types d'entrées possibles, mais elle fait perdre des informations structurelles dès le départ.

1.2 Les différentes solutions

J'ai choisi la deuxième méthode pour sa généralité, son uniformité, et parce qu'elle permet d'introduire des caractéristiques locales de la surface, sa topologie. Il faut également être capable de répondre aux questions suivantes :

- Que signifie mailler une surface?
- Quel rapport avec le maillage d'un ensemble de points extraits d'une surface?

Ces questions anodines permettent de se rendre compte de la difficulté du problème, mais surtout de celle de sa formulation exacte. En effet, nous manipulons ici 3 objets distincts: la surface, le maillage, et les points digitalisés sur la surface.

Je me restreins à l'étude des maillages passant par les points digitalisés. Certains ont déjà travaillé sur l'extension consistant à choisir d'autres points comme sommets du maillage, afin d'obtenir un maillage plus homogène, ou bien dont la densité varie en fonction de la courbure de la surface [Tu92] [BG].

Les points sont inclus dans la surface et dans le maillage, mais on constate tout de suite qu'il existe une infinité de surfaces ainsi qu'un grand nombre de maillages passant par un ensemble de points donnés.

Pour trouver le maillage à partir des points et de la surface, lorsque celle-ci est connue, on bénéficie d'une indication topologique qui permet de donner un sens à la notion de maillage correct, en fixant par exemple comme contrainte que le maillage doit être le plus "proche" possible de la surface (écart entre la surface et le maillage de volume minimal, par exemple). Cela dit, il subsiste toujours des cas où l'on ne peut pas déterminer clairement ce qu'il faut faire, parce que la densité des points n'est pas assez importante, par rapport à la courbure locale de la surface.

Je ne me risquerai donc pas à donner une définition à la notion de maillage optimal.

Lorsque l'on dispose uniquement des points, et d'aucune information topologique, il faut reconstruire, en même temps que le maillage, certaines caractéristiques locales de la surface. Or, comme mentionné précédemment, il n'existe pas de surface unique passant par un ensemble de points donnés. Il faut donc se fixer des critères sur une surface "plausible", et engendrer un maillage qui corresponde à ce qui aurait été produit par un algorithme connaissant aussi la surface.

Les critères déterminants si une surface correspond bien à un ensemble de points ne peuvent pas être fondés sur des bases intrinsèques, il faut donc utiliser des arguments comme, par exemple :

- La courbure de la surface est bornée (reste à définir la borne), sauf dans le cas d'une arête vive.
- Un trou dans un objet ne peut pas avoir un diamètre inférieur à telle valeur.

Pour mailler des points non structurés, on peut par exemple (idée de Jean-Daniel Boissonnat) se ramener au cas des coupes parallèles, en projetant les points sur le plan de coupe le plus proche.

Une autre approche, celle des alpha-shapes, décrite dans [Ed], permet de mailler tous les triangles qu'une boule de rayon α ne peut pas traverser (dans le cas contraire, ce sont des trous), moyennant une densité uniforme des points sur la surface. On peut calculer les maillages correspondants à toutes les valeurs de α en temps quadratique. Cette méthode possède le défaut important de nécessiter une densité uniforme.

J'ai préféré approcher le problème de manière plus topologique et locale, car c'est à mon avis un point de vue qui permet de traiter une bonne quantité de cas particuliers, qui gênent les algorithmes classiques, qui, dans l'ensemble, partent du tout pour arriver au détail. On arrive ainsi à gérer les arêtes vives, les variations de densité de points, etc...

1.3 Plan général de l'algorithme

Notons que je n'utiliserai dans la suite, sauf mention contraire, que la distance "infinie" [2], de sorte que les boules sont en fait des cubes alignés selon les axes principaux. Cela permet des calculs plus rapides qu'avec la distance euclidienne, tout en sachant que ces deux métriques sont équivalentes. De plus, je supposerai que dans l'ensemble de points fournis, il n'en existe pas deux ayant les mêmes coordonnées.

L'algorithme procède comme suit :

- Dans un premier temps, on structure les points de manière à pouvoir déterminer rapidemment les plus proches voisins d'un point quelconque. La structure choisie (un "arbre octal" ou "octree") est arborescente, de taille linéaire, et se construit en temps $O(N \log N)$. Le temps de recherche dépend de la répartition des points, mais est proportionnel à la hauteur de l'arbre, c'est à dire, en moyenne, logarithmique par rapport au nombre de points.

- On estime ensuite une distance caractéristique de l'ensemble, d, qui est la moyenne des distances au plus proche voisin, pour quelques points choisis au hasard. A noter que les deux premières étapes sont permutables, on peut d'ailleurs se servir de l'une pour l'autre, et réciproquement.

- L'idée générale est la suivante : on initialise le maillage, puis on l'étend progressivement le long de sa frontière (liste[s] de points), en considérant un point de cette frontière, ses plus proches voisins, une estimation du plan tangent local, et en essayant d'étendre le maillage (donc la frontière) en respectant des règles de base, qui sont censées garantir

2. La distance "infinie" entre deux points est le maximum des valeurs absolues des différences de chaque coordonnée.

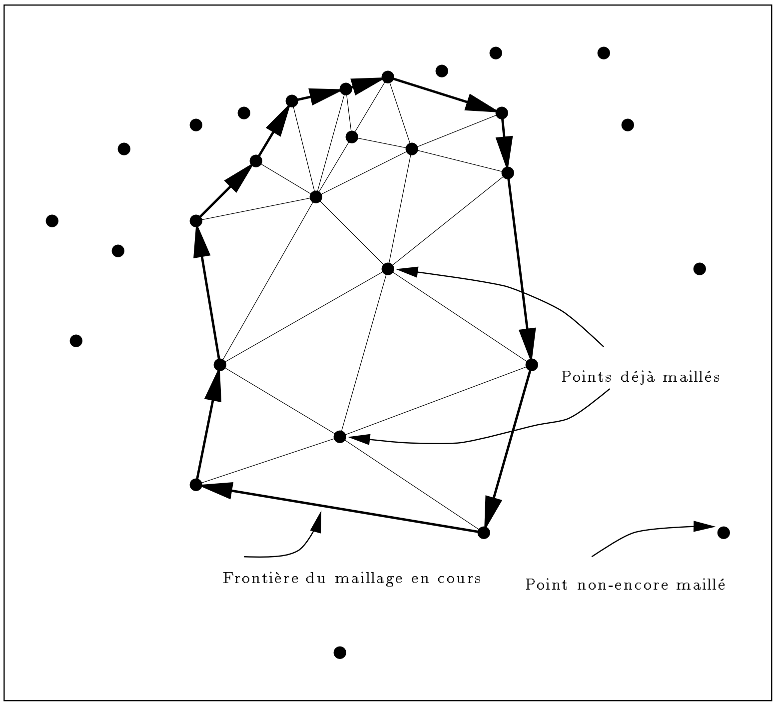

FIG. 1 – *Schématisation d'un maillage en cours d'élaboration.*

un maillage global correct. A noter que la frontière, au cours de l'évolution de l'algorithme, pourra se scinder et se recoller. Cela permet de considérer des surfaces non connexes, de genre quelconque, fermées ou non (figure 1).

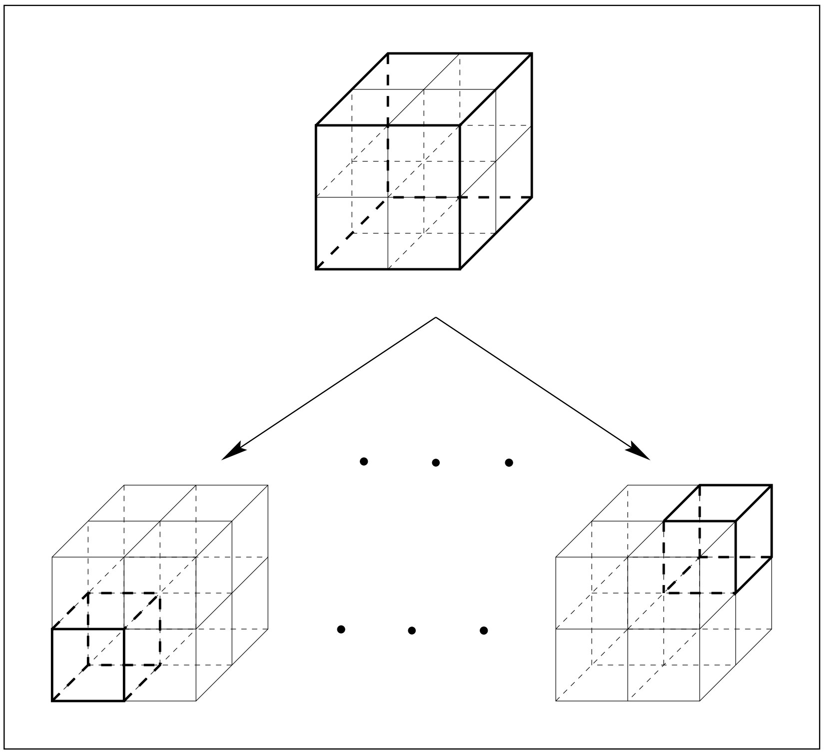

FIG. 2 – *Arbre octal.*

2 Algorithme

2.1 Structures utilisées

En entrée, les points sont fournis par un tableau de coordonnées.

2.1.1 Localiser rapidemment les points de l'ensemble situés dans un voisinage d'un point donné

La structure de localisation est la suivante : on considère un arbre, dont chaque sommet possède huit fils. Chaque sommet représente un cube de l'espace, subdivisé en 8 cubes de même taille (figure 2). Le cube "racine" englobe donc tous les points.

Il y a deux possibilités pour les feuilles :

– On ne limite pas la hauteur de l'arbre afin de n'avoir qu'un seul point par feuille.

– On borne la hauteur de l'arbre, ce qui correspond à minorer la distance de séparation. Chaque feuille contient alors une liste de points. Ces listes ne doivent pas être trop grandes.

J'ai choisi de programmer la première solution, parce qu'elle est plus uniforme, et bien qu'on ne puisse pas déterminer précisément la hauteur de l'arbre dans ce cas, on peut avoir une idée relativement bonne de la distance moyenne des feuilles à la racine.

Cette structure est connue sous le nom d'arbre octal, ou octree. Si les points sont répartis de manière suffisamment homogène, la taille de l'arbre peut être estimée comme suit : si on dispose de N points sur une surface, et qu'ils sont équitablement répartis (distance moyenne entre deux points voisins d, avec une répartition ne donnant pas de points très rapprochés), on obtient une surface qui peut tenir dans un cube de côté $O(d\sqrt{N})$. Le plus petit sous cube aura alors la taille d, donc la hauteur de l'arbre sera $O(log_2(\sqrt{N})) = O(logN)$. La taille mémoire occupée par cette structure est linéaire en fonction du nombre de points.

Le calcul de cette structure s'effectue par un algorithme incrémental, une fois que le cube englobant est déterminé. Il suffit simplement de faire descendre chaque nouveau point dans le sous-cube qui le contient, et ceci récursivement.

Pour déterminer les points apartenant à un cube donné quelconque (un point et une distance déterminent une boule dans la distance infinie), il suffit de faire un parcours en profondeur de l'arbre, en s'arrêtant lorsqu'un cube de l'arbre a une intersection vide avec le cube donné. Ainsi, on ne parcourt pas les sous-cubes qui n'ont aucune chance d'avoir un point commun avec le cube donné. Le temps de calcul dépend de la taille du cube par rapport à d et du nombre de points que l'on récupère ainsi. Le but étant d'utiliser cette structure pour trouver les points voisins, ces deux valeurs seront petites, et on peut s'attendre à un comportement moyen proportionnel à la hauteur de l'arbre.

A noter qu'il y a aussi la solution de calculer une triangulation de Delaunay, pour localiser rapidemment les points. La solution des octrees est beaucoup plus simple à programmer, et compte tenu du problème, ne doit pas être plus lente que la méthode utilisant la triangulation de Delaunay.

2.1.2 Structure utilisée pour calculer le maillage

Comme décrit brièvement dans la section précédente, le calcul s'appuie sur l'extension d'une frontière. J'ai choisi un codage pratique, ce qui est

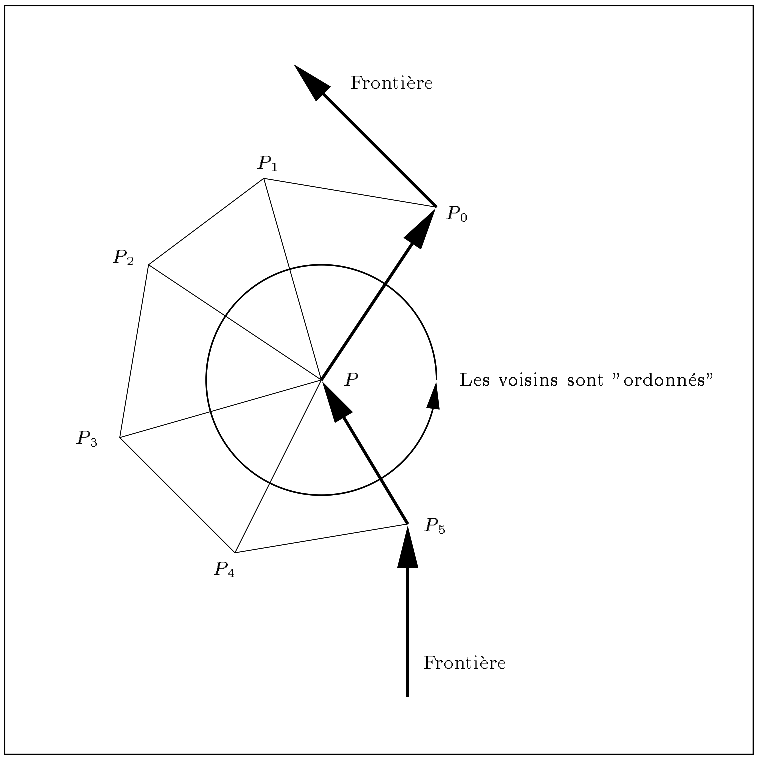

FIG. 3 – *Structure locale du maillage.*

indispensable si l'on veut traiter de grands ensembles de points. En conséquence, j'utilise un tableau, qui stocke, pour chaque point, deux pointeurs :

- Le premier détermine si le point a déjà été maillé, et si oui, pointe vers une liste ordonnée (c'est important pour des raisons de performance) de ses voisins sur le maillage (figure 3). La liste en question est une liste classique (bien qu'une liste bouclée puisse être employée, peut-être même avantageusement) de pointeurs vers les coordonnées des points. L'ordre sur la liste doit permettre de reconnaître rapidement si 3 points forment un triangle du maillage, donc deux points consécutifs sur la liste (modulo le bouclage de la liste), soit forment un triangle avec le point dont provient la liste, soit sont sur la frontière.

- Le second détermine si le point est sur la frontière, et si oui, pointe vers le point suivant sur la frontière (qui est donc orientée). On note immédiatement que ceci impose la restriction suivante : la frontière ne

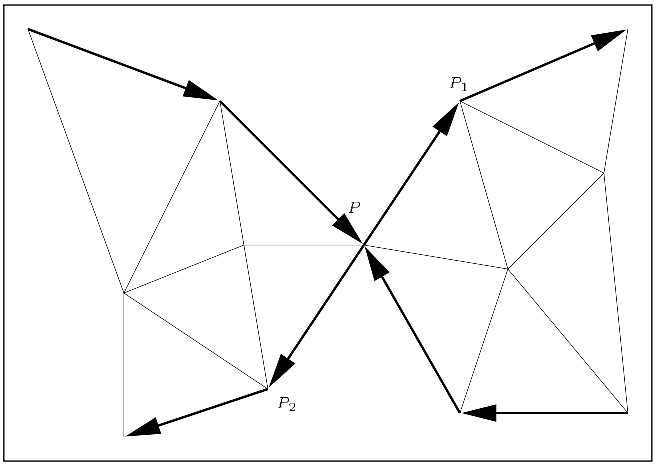

FIG. 4 – *Il existe plusieurs points sur la frontière (P_1 et P_2) suivant P, le point central.*

peut passer qu'une seule fois par un point donné (figure 4). Ceci est assez contraignant, si l'on veut une frontière suffisamment "souple", qui puisse se découper et se recoller (ce qui permet de mailler des surfaces un tant soit peu compliquées). Notons que comme la frontière est orientée, on s'interdit de mailler entièrement un ruban de Moebius (figure 5), car lors du recollement final (qui scinde en fait la frontière), les deux parties qui se feront face seront orientées dans le même sens. De plus amples explications seront données dans les paragraphes suivants, sur la gestion de la frontière. Ce choix devrait néanmoins permettre de mailler correctement les surfaces provenant vraiment d'objets 3D.

2.2 Eléments topologiques

C'est ici que je définis des critères permettant de se forger une idée plus précise sur la forme de la surface, donc sur ce qu'il faut et ne faut pas mailler.

Le critère topologique de base est la détermination du plan tangent à la surface en un point de l'ensemble. On peut approximer sa valeur en quelques étapes :

- Grâce à la structure de localisation, on recherche les points de l'ensemble qui sont à une distance inférieure à un certain seuil δ du point choisi, proportionnel (3 ou 4 fois plus grand, mais pas plus) à la distance caractéristique d calculée au départ.

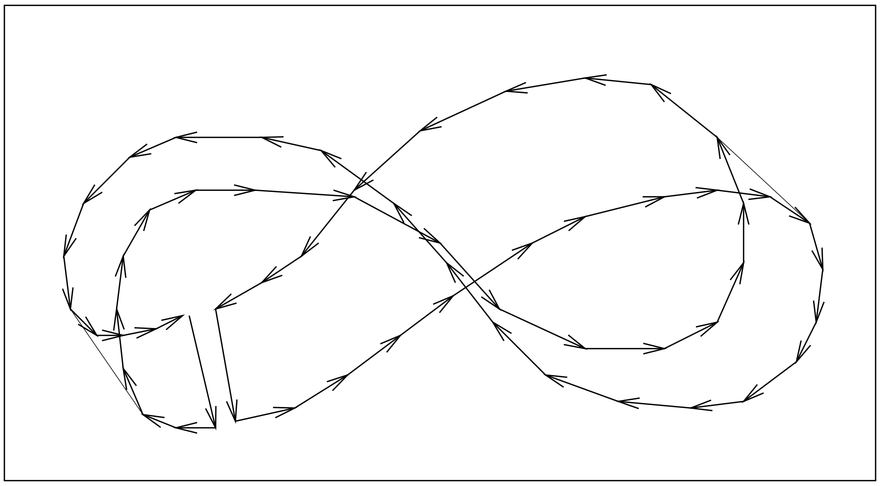

FIG. 5 – *Le maillage du ruban de Moebius ne peut pas être terminé...*

- On calcule ensuite le vecteur normal correspondant au plan tangent "des moindres carrés". Ce calcul peut avantageusement être coefficienté, pour chaque point, avec l'inverse du carré de la distance au point central choisi, ce qui retire du poids aux points les plus éloignés.
- On élimine ensuite les points qui font un angle trop faible avec la normale, et on peut éventuellement effectuer un calcul plus précis de la normale, avec les points restants. (figure 6)

Ce que je n'ai pas fait dans cette première version de mon programme, mais qui me paraît utile, c'est de précalculer toutes les normales, plutôt que de les recalculer au vol. En effet, cela permet, pour un coût qui finalement s'avère raisonnable, d'affiner par lissage (par exemple), la topologie de la surface. Cette méthode permettrait également de calculer les courbures et les arêtes vives.

Grâce à la valeur estimée du plan tangent à la surface, on peut se prémunir (plus ou moins efficacement) contre le maillage des replis (figure 6), car on peut éliminer des points locaux ceux qui se trouvent trop éloignés du plan tangent (hors d'un cône limite). Plus théoriquement, en se basant sur une courbure maximum de la surface, on pourrait éliminer les points situés en dehors de deux paraboloïdes (figure 7) placés de part et d'autre du plan tangent, mais ceci est entaché par l'imprécision du calcul du plan tangent, et par le fait que l'échantillonage n'est garanti qu'à une certaine précision (celle des capteurs).

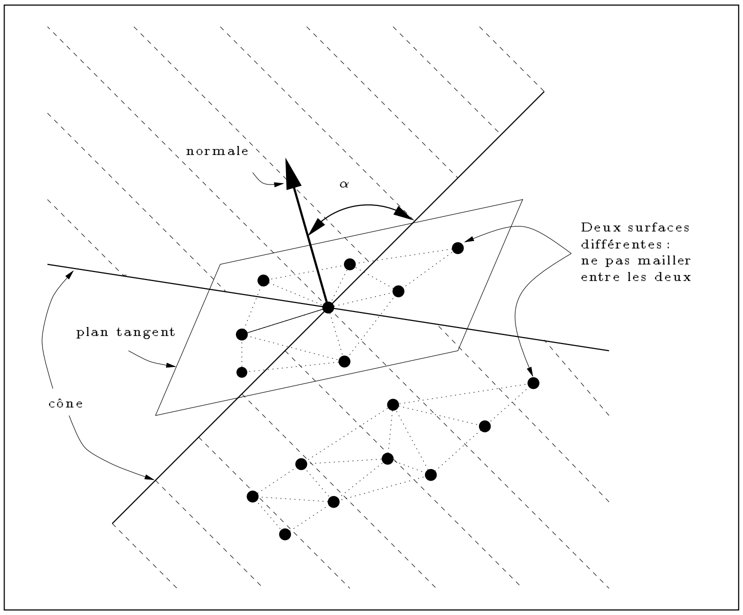

FIG. 6 – *Comment gérer les surfaces qui se touchent presque.*

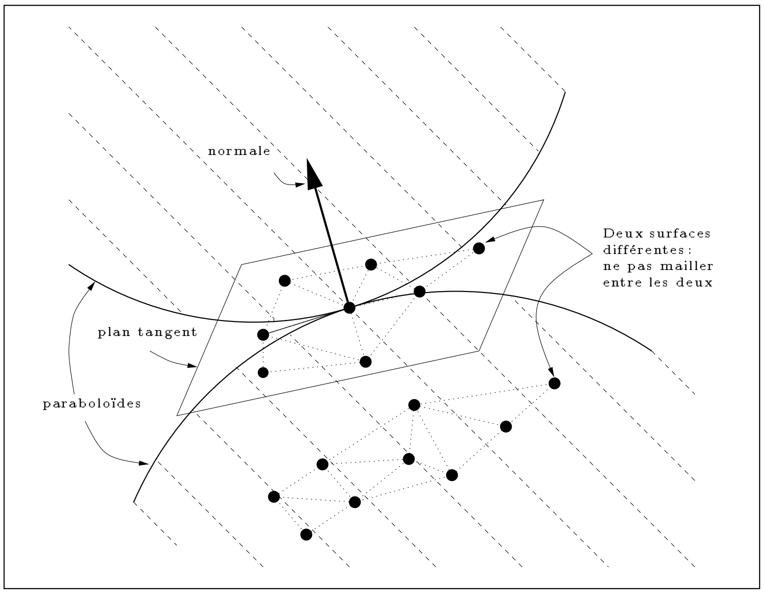

FIG. 7 – *En théorie, on peut délimiter les zones avec des paraboloïdes.*

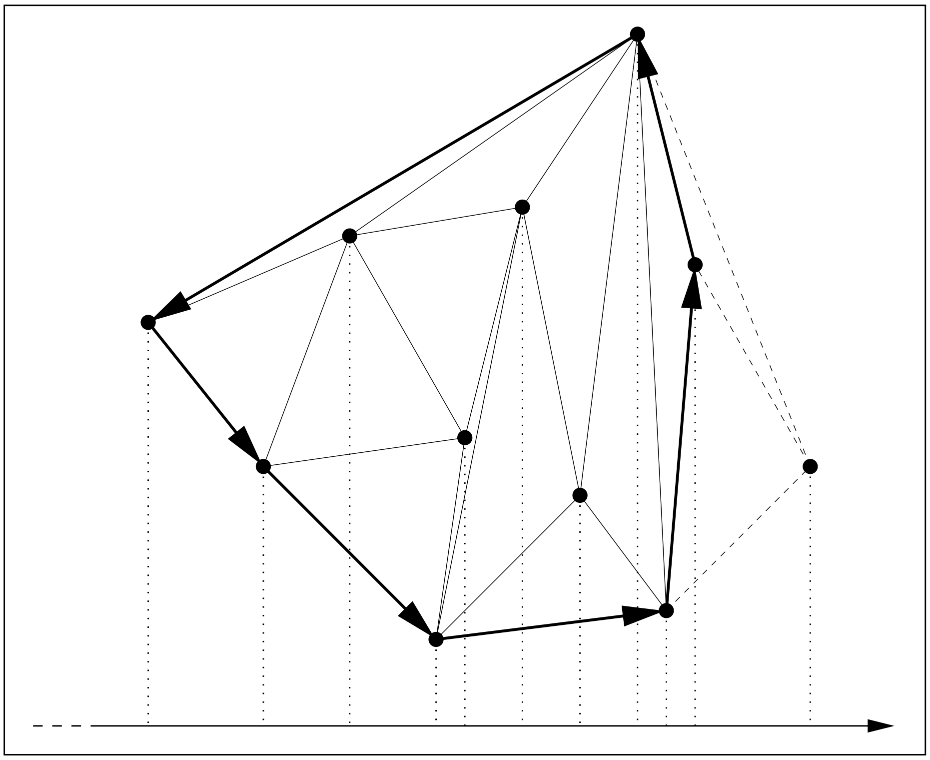

FIG. 8 – *Maillage initial calculé simplement par balayage.*

2.3 Initialisation du maillage

On initialise le maillage par quelques points voisins (4 ou plus), en les projetant dans le plan tangent local estimé, et en triangulant leurs projetés dans ce plan. On défini la frontière initiale comme l'enveloppe convexe de cette triangulation.

La méthode utilisée pour réaliser ce premier maillage n'a pas besoin d'être très élaborée, puisqu'elle est limitée au minimum de points possibles. J'ai choisi un algorithme par balayage, pour sa simplicité (figure 8).

Il est à noter que l'on peut initialiser le maillage en plusieurs endroits et à plusieurs moments, c'est ce qui permet de mailler des surfaces non connexes.

On peut également initialiser un maillage à un endroit, lorsque l'on ne peut plus agrandir le maillage courant. C'est typiquement ce qui peut se passer dans le cas d'objets non connexes, ou à plusieurs faces, lorsque le maillage s'arrête sur une arête vive (il faut pour cela être capable de détecter ces arêtes).

On choisit ensuite une distance caractéristique de l'objet, qui va fixer, pour un temps, la taille du voisinage des points. Cette valeur augmentera au cours de l'algorithme.

2.4 Comment étendre le maillage

C'est le point délicat de l'algorithme, et le plus difficile à implanter. Je vais décrire la méthode que j'ai programmée.

Tout d'abord, il faut trouver un point qui se situe sur la frontière. Pour se faire, on peut parcourir l'ensemble des points, et prendre le premier qui convient, ou maintenir l'ensemble des points de la frontière. Faute de temps, je n'ai pour le moment implanté que la première solution, mais il va de soi que la deuxième est meilleure.

On détermine ensuite ses voisins, et on élimine ceux qui sont supposés être sur un autre pli de la surface. Parmi tous ces points, certains sont maillés, d'autres pas, et d'autres encore sont sur la frontière. Je les projète tous dans le plan tangent, ce qui est censé refléter la même situation, mais qui pose parfois des problèmes (figure 12).

Je procède ensuite par une boucle qui considère l'extension optimale du maillage dans la configuration actuelle, met à jour le maillage (frontière incluse), puis recommence jusqu'à ce qu'il ne reste plus de point local à mailler, ou bien jusqu'à ce qu'un autre évènement empêche de continuer.

L'extension consiste en un nouveau triangle qui vient se coller à la frontière (figure 9). L'optimalité est définie comme la maximalité du plus petit angle du nouveau triangle ainsi créé. On obtient ainsi un nouveau triangle qui est le moins plat possible. Par là même, il a de fortes chances de "coller" davantage à la surface.

Il se peut (et j'en tiens compte dans mon programme) qu'un échange ("triangle flip" ou encore "edge swapping") soit avantageux, c'est-à-dire que sur deux triangles adjacents (dont le nouvellement formé), formant un quadrilatère convexe, le choix de l'autre diagonale comme séparation fournisse un plus petit angle plus important, dans une configuration que dans l'autre, on a intérêt dans ce cas à faire le changement (figure 10).

Cependant, ce qui est trivial dans le plan l'est moins dans l'espace : la notion de quadrilatère convexe n'existe pas, j'ai considéré que c'est ce que j'obtenais en projetant, mais cela peut mener à de gros problèmes d'incohérence du maillage, car certaines configurations sont sensibles au choix de la normale, si fines soient les différences (figure 12). On peut obtenir aussi de meilleures approximations des arêtes vives, en tenant compte de la courbure locale (figure 11).

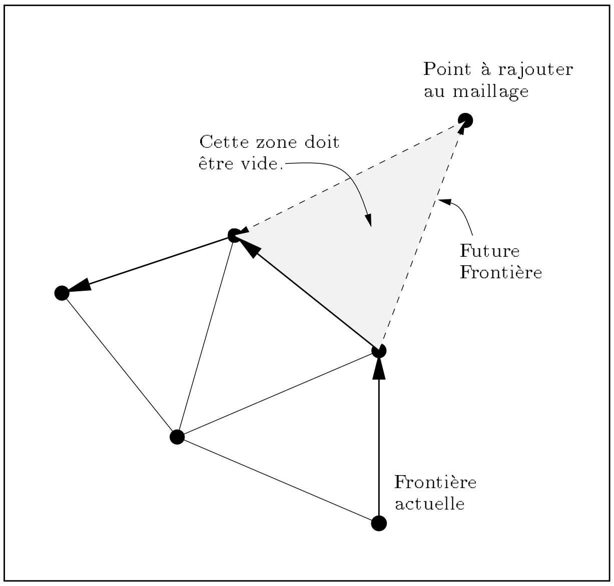

FIG. 9 – *Extension du maillage.*

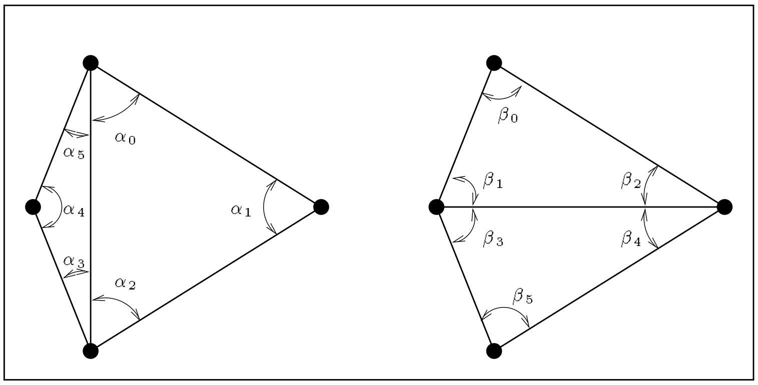

FIG. 10 – *"Triangle flip" :* $\min_i \alpha_i < \min_i \beta_i$

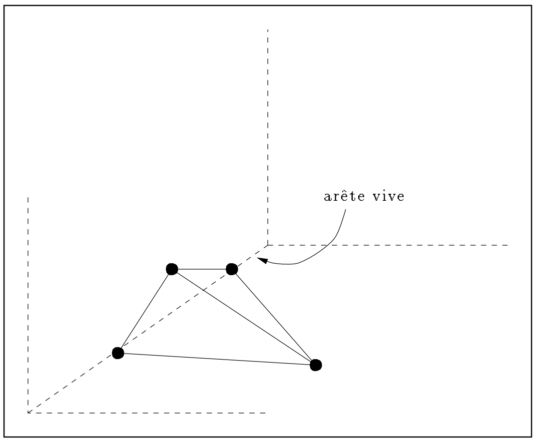

Fig. 11 – *Le "Edge swapping" à manier avec précaution en 3D. Il conviendrait de faire intervenir la courbure.*

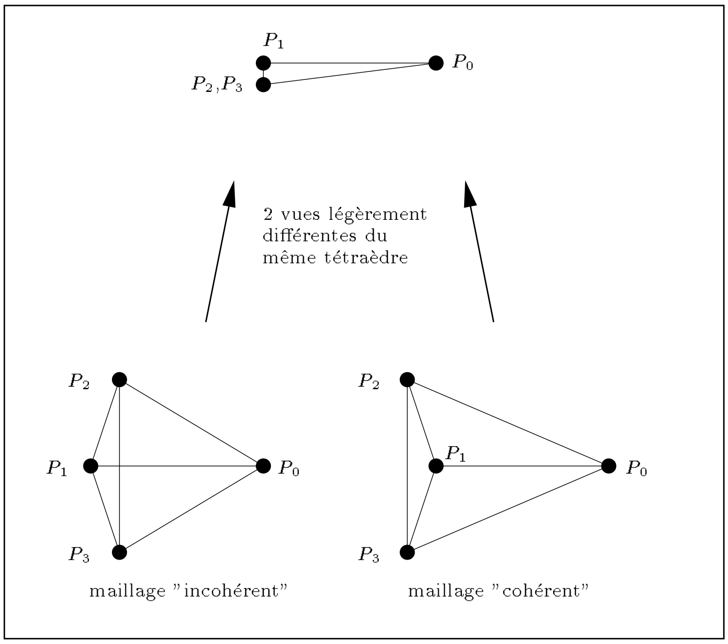

Fig. 12 – *Danger de deux vues légèrement différentes.*

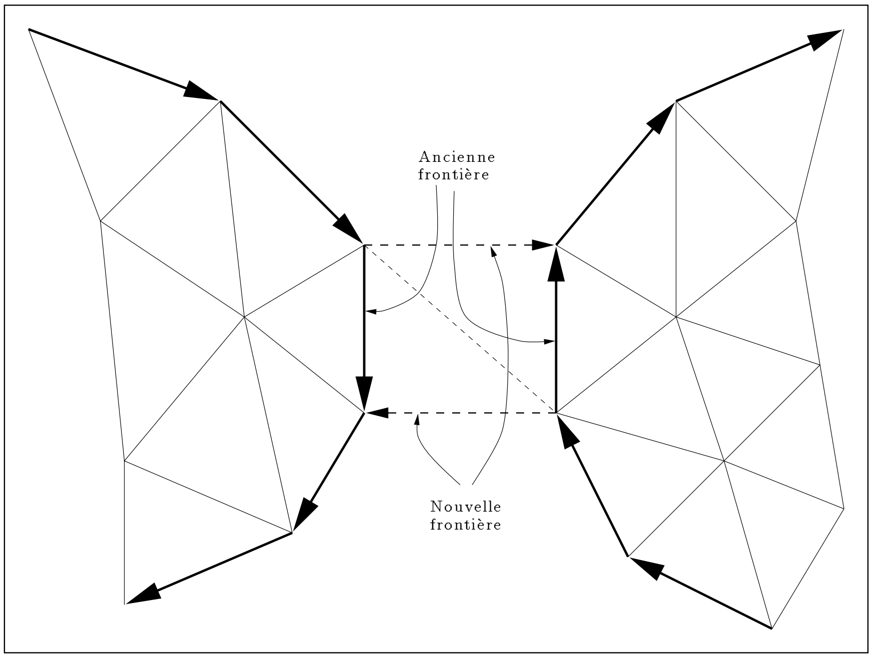

FIG. 13 – *Deux morceaux de frontière correctement orientés se rejoignent.*

L'extension doit vérifier des critères de base, pour être la plus cohérente possible, il y a toujours des cas où des problèmes peuvent surgir... Plusieurs d'entre eux me sont apparus lors des essais pratiques.

– Il ne doit pas y avoir d'autre point à l'intérieur d'un nouveau triangle.

– Aucun segment ne doit couper le nouveau triangle.

– On peut faire une extension avec un point déjà maillé, s'il est sur la frontière. Cela correspond à un recollement (ou découpage) de la frontière (figure 13).

– D'autres cas moins importants, bien qu'étant indispensables au bon déroulement de la suite de l'algorithme.

Une fois que l'on ne peut plus étendre le maillage en local, on recommence avec un autre point de la frontière. Jusqu'à temps que l'on ne puisse plus étendre nulle part.

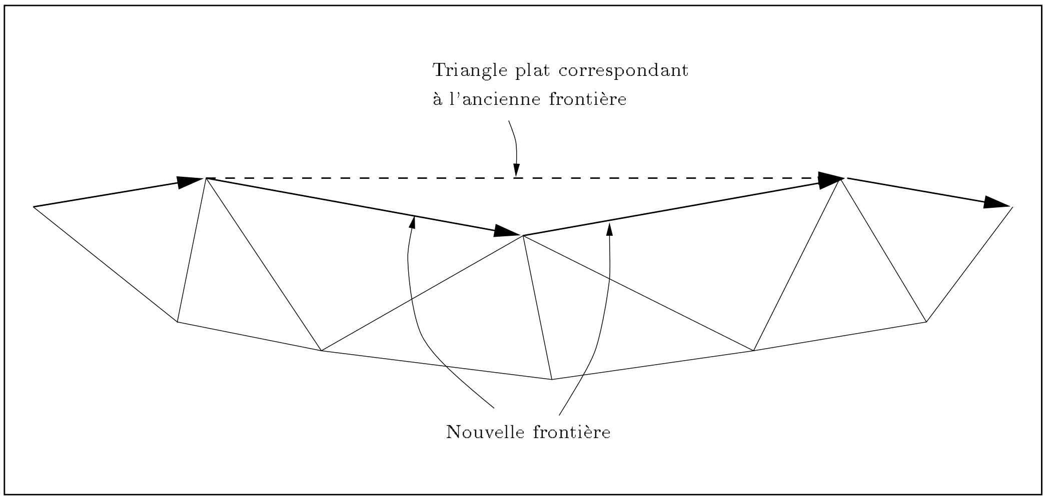

FIG. 14 – *La frontière était formée de triangles trop plats.*

2.5 Passes d'optimisation

Lorsque l'on ne peut plus étendre le maillage, on procède à diverses optimisations :

- Une ou plusieurs passes sur toutes les arêtes du maillage en cours, et test s'il est préférable d'échanger des "diagonales de quadrilatères". (même méthode que précédemment)
- Ce type de critères peut être étendu à des "refontes" locales plus importantes, mais plus délicates à mettre en oeuvre. Je n'en ai implanté aucune à l'heure actuelle.
- La frontière a tendance à être trop "convexe", i.e. elle est faite de longs triangles plats. Ils ne facilitent pas les extensions, donc j'essaie d'en retirer, selon un critère de platitude des triangles (figure 14).
- La frontière a parfois des "trous", qu'il convient de reboucher (figure 15).

Puis, je réessaye d'étendre le maillage, après avoir augmenté légèrement la distance d qui fixe le voisinage des points. Cette technique a plusieurs avantages :

- Partir d'une petite distance permet de ne pas avoir trop de points dans les voisinages, ce qui accélère l'algorithme. De plus, il ne faut pas

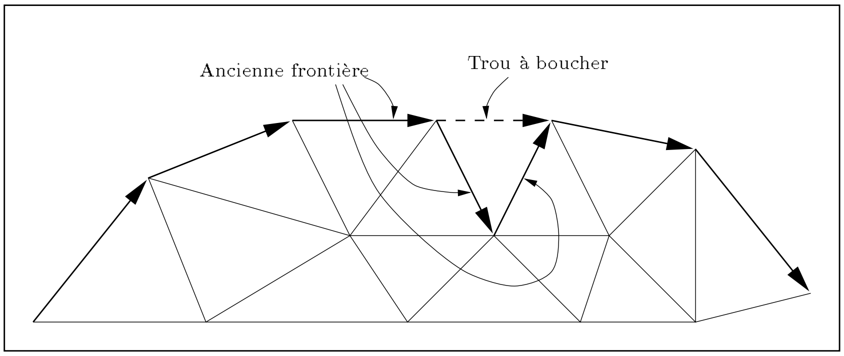

FIG. 15 – *La frontière contient des trous.*

oublier que l'on doit se baser sur des observations locales.

– L'algorithme s'arrête sur des "frontières naturelles", qu'il peut passer si et seulement si on augmente la distance de voisinage. Ces limites peuvent être des zones moins denses, ou bien des arêtes vives.

Dans tous les cas, il faut arrêter le maillage assez tôt, pour éviter de mailler des points trop éloignés. Ce paramètre dépend fortement du type d'objet, puisque c'est un paramètre global (qui s'applique à l'ensemble de la pièce), il me paraît même difficile de le calculer automatiquement.

2.6 Fin de l'algorithme

L'algorithme prend fin lorsque l'on ne peut plus mailler, et que l'on a atteint la limite fixée, pour la distance d (pour les voisinages des points).

2.7 Etude de complexité

La complexité globale s'établit à $O(N \log N)$, pour une entrée de N points, sous réserve d'une répartition des points fournissant un octree de hauteur logarithmique. Le facteur de temps dépend surtout de la programmation de l'extension du maillage, et de la gestion de la frontière.

2.8 Intéractivité optionnelle

On peut demander à l'utilisateur certains paramètres, qui peuvent aider l'algorithme, bien qu'il n'en ai a priori pas besoin. Ce sont :

- Le ou les points de départ, pour l'initialisation du maillage. On peut suggérer des zones plus denses que d'autres.
- La distance initiale, déterminant les voisinages.
- Le moment où l'on arrête d'augmenter cette distance.

Rien de tout cela n'est implanté dans mon programme.

3 Conclusion

3.1 Résultats expérimentaux

L'algorithme a une complexité estimée à $O(N \log N)$, N étant le nombre de points en entrée. Le temps de calcul s'établit à 7 minutes 30 pour 14000 points sur un Pentium 166MHz sous Linux 2.0.

Mon programme n'étant pas encore tout à fait terminé, ce ne sont que des résultats partiels, et pour le moment non-satisfaisants du point de vue de la qualité du maillage. Cela dit, une interface graphique que j'ai développée, me permet de visualiser, et de faire évoluer sous tous les angles, le résultat de mon algorithme sur des entrées aléatoires ou digitalisées.

Il reste encore quelques bugs à corriger, et quelques fonctions à intégrer (par priorité décroissante) :

- gérer le cas où on donne deux points égaux en entrée (ce qui dans l'état actuel du programme produit une récursivité infinie),

- gérer correctement des optimisations (qui sont inactivées pour le moment, sous peine de maillage incorrect),

- gérer le "edge-swapping" pendant la recherche de l'extension optimale,

- gérer une file de priorité pour trouver un point de la frontière supposé intéressant le plus rapidemment possible (ceci influe fortement sur la complexité pratique de l'algorithme),

- lorsque tout ceci fonctionnera correctement, essayer d'optimiser l'implantation de l'algorithme.

Tout ceci devrait prendre deux à trois semaines de travail. Je compte aussi profiter de l'occasion pour programmer une méthode simple de maillage des grilles (détection des trous par hystérésis).

3.2 Améliorations souhaitables

Le précalcul des normales à la surface en chaque point me paraît utile, ainsi que tout ce qui peut ajouter d'informations topologiques. Ceci est essentiel, car une meilleure connaissance dans ce domaine, permet d'éviter les erreurs qui semblent abérantes. En particulier, la gestion des courbures et des arêtes vives me paraît une extension intéressante.

3.3 Remerciements

Je remercie l'équipe qui m'a acceuilli au sein de MDTV, tout particulièrement Nazih Saadé et Xavier Benveniste. Je tiens également à dire merci aux membres du projet Prisme de l'INRIA Sophia, qui ont pris le temps d'écouter et critiquer mon idée, et de me fournir des références sur le sujet, en particulier Jean-Daniel Boissonnat et Stéphane Nullans.

3.4 Inspiration

La bibliographie qui suit m'a permis de constater que le domaine intéresse de nombreux chercheurs et industriels. Je pense cependant que l'approche concernant le maillage surfacique d'un ensemble de points non structurés est assez originale. J'ai souligné les points pouvant se rattacher au sujet précis, dans quelques références.

Références

[BY95] *Géométrie Algorithmique*, Boissonnat, Yvinec, 1995.

[At] *Thèse*, D. Attali, TIMC Grenoble.

[Ch] *Modélisation géométrique par vision artificielle*, thèse de Xin Chen.

[CF94] *Surface modelling of range data by constrained*, Xin Chen, Francis Schmitt, CAD vol 26, Août 94.

[CI91] *Les arbres octaux*, Bulletin technique de CISI Ingenierie, Janvier 91.

[CA93] *Evaluation of a Delaunay-based method for surface approximation*, CAD vol 25, Novembre 93.

[BG] *Encore Delaunay*, Borouchaki et George, Rapport de Recherche Inria No 2461.

[DH] *Anisotropic Surface Mesh Generation*, Castro Diaz et Hecht, Rapport de Recherche Inria 2672.

[CA88] *Triangulation of scattered data in 3D space*, Choi, Shin, Yoon, Lee, CAD vol 20, Juin 88.

[Bo84] *Geometric Structures for Three-Dimensionnal Shape Representation*, Boissonnat, 1984.

[Ed] *Three-dimensional Alpha Shapes*, Herbert Edelsbrunner and Ernst P Mücke.

Cet article traite des α-shapes, c'est-à-dire des triangulations obtenues à partir d'un nuage de points, sur lequel on fait "rouler" une boule de rayon α. Si la boule a un rayon infini, on obtient l'enveloppe convexe, et plus le paramètre α diminue, plus on obtient de trous dans la surface. Un algorithme est proposé pour obtenir en temps quadratique les triangulations correspondant à toutes les valeurs d'α. Le principal inconvénient de cette technique est qu'elle ne s'adapte pas à la densité locale des échantillonnages de la surface, c'est-à-dire que le paramètre α est uniforme sur toute la surface.

[SL95] *A General Surface Approach to the Integration of a Set of Range Views*, Marc Soucy and Denis Laurendeau, IEEE Transactions on pattern analysis and Machine Intelligence, Vol. 17, NO. 4, April 1995.

Cet article traite des recollements de diverses prises de vue déjà triangulées. Il ne m'intéresse donc pas directement, mais peut-être utile pour une autre approche, typiquement pour des maillages par grilles.

[SZ92] *Decimation of Triangle Meshes*, William J. Schroeder, Jonathan A. Zarge, William E. Lorensen, Computer Graphics, 26, 2. July 1992.

Cet article traite d'un algorithme permettant d'obtenir une triangulation plus ou moins dense d'une surface déjà triangulée. On peut visionner ainsi plus rapidement un objet (en face cachées), par exemple. Ce n'est pas en rapport direct avec mon travail, mais cet algorithme pourrait avantageusement être intégré à un des logiciels développés par MTDV.

[Tu92] *Re-Tiling Polygonal Surfaces*, Greg Turk, Computer Graphics, 26, 2. July 1992.

C'est un article intéressant, qui se propose de fournir, à partir d'un maillage grossier d'une surface (par exemple un maillage passant par des points digitalisés), un maillage où les triangles sont plus réguliers (plus équilatéraux). On peut aussi paramétrer la taille des triangles. L'algorithme fournit d'excellents résultats, à en juger par les photos incluses dans l'article.

A noter que Paul-Louis GEORGE fait à peu près la même chose.

[LM95] *Surface Approximation of a Cloud of 3D Points*, Chia-Wei Liao and Gérard Medioni, Graphical Models and Image Processing, Vol. 57, No. 1, January 1995.

Cet article traite un problème similaire au nôtre, le but étant ici de couvrir la surface par des rectangles. Il ne nous concerne donc pas directement, bien que des idées soient communes. L'approche générale ne me paraît néanmoins pas la bonne, dans le cas précis qui nous occupe.

[Ho92] *Surface Reconstruction from Unorganized Points*, Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle. Computer Graphics, 26, 2. July 1992.

C'est de loin l'article qui correspond le plus à nos besoins, il contient un grand nombre d'idées communes avec ce que j'avais préparé, bien que le raisonnement ne soit pas poussé suffisamment (peut-être par

souci de simplicité). En particulier, le cas de deux surfaces "se frôlant" n'est pas traité, et l'algorithme tel quel ne devrait pas s'en sortir.