Académie de Montpellier
Université Montpellier II
Sciences et **T**echniques du **L**anguedoc

Thèse

présentée au Laboratoire d'Informatique de Robotique
et de Microélectronique de Montpellier pour
obtenir le diplôme de doctorat

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Information, Structures, Systèmes**

Synchronisation pour l'insertion de données dans des maillages 3D.

par

Nicolas Tournier

Version du 14 octobre 2014

Directeur de thèse
M. William Puech, Professeur ..... Université Montpellier II, LIRMM, Montpellier, France

Co-Directeur de thèse
M. Gérard Subsol, Chargé de Recherche ..... CNRS, LIRMM, Montpellier, France

Rapporteurs
M. Marc Antonini, Directeur de Recherche ..... CNRS, I3S, Sophia Antipolis, France
M. Florent Dupont, Professeur ..... Université Lyon I, LIRIS, Lyon, France

Examinateur
M. Jean-Marc Chassery, Directeur de Recherche..... CNRS, GIPSA-lab, Grenoble, France

Table des matières

Bibliographie **113**

Introduction

Contexte de la thèse

Cette thèse s'inscrit dans le cadre d'un contrat CIFRE entre le Laboratoire d'Informatique, de Robotique, et de Microélectronique de Montpellier (LIRMM, UMR 5506 CNRS, Université de Montpellier 2) et la société STRATEGIES S.A. située à Rungis.

STRATEGIES est une entreprise de développement de logiciels qui produit la suite Roman CAD Software (RCS) spécialisée dans le design 2D/3D de matériaux souples à destination de l'industrie de la mode. En particulier l'industrie de la chaussure, où STRATEGIES propose une solution numérique à toutes les étapes du processus de création.

Pour fabriquer une chaussure, il faut d'abord créer la forme intérieure. Cette partie est réalisée par un formier qui sculpte cette forme traditionnellement sur du bois (figure 0.1). Par la suite le design extérieur de la chaussure est réalisé. Les matériaux, les couleurs, les formes extérieures sont ainsi choisis.

Que ce soit la création de la forme ou le design extérieur, ces phases sont prises en compte dans la suite logicielle développée par STRATEGIES (figure 0.2). Il est également possible d'importer la forme digitalisée pour que les designers travaillent ensuite numériquement sur le design extérieur.

FIGURE 0.1 – Forme de chaussure en bois. Source : http ://www.romans-cad.com

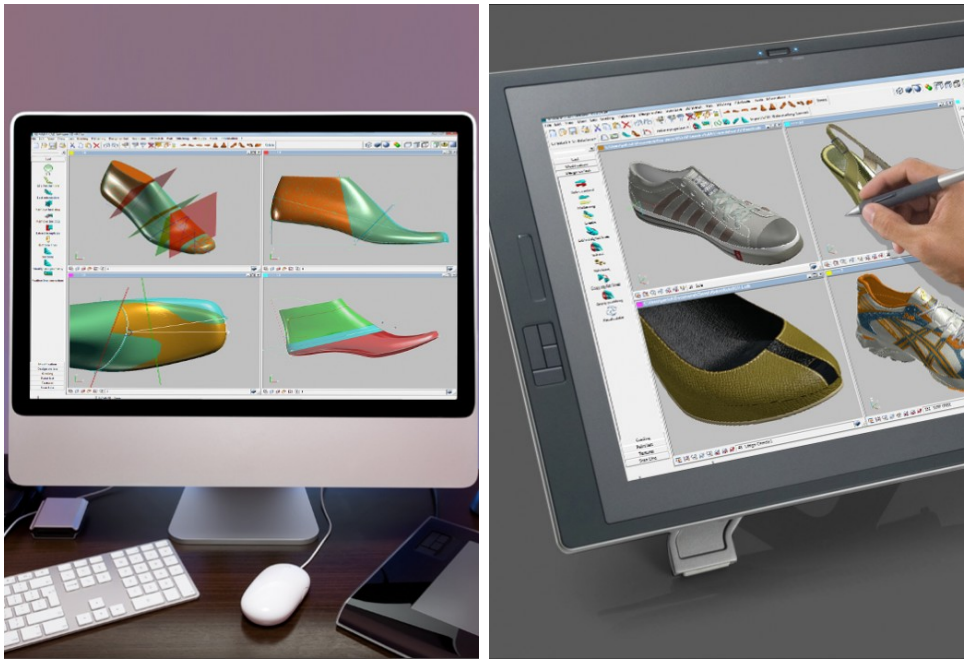


FIGURE 0.2 – Roman CAD Software. Source : http ://www.romans-cad.com

La problématique de la synchronisation des données.

Tout comme une musique, un film, une image, une application, l'industrie de la mode veut protéger ses créations numériques. Dans notre cas, deux "objets" ont besoin d'être protégés : la forme intérieure numérisée et le design extérieur.

A partir d'une forme de créateur, il est facile d'en obtenir une nouvelle similaire ; en faisant un simple lissage, par exemple. Le problème est donc de savoir si la nouvelle forme créée est originale ou issue d'une forme de créateur et de pouvoir identifier le propriétaire. Pour cela le tatouage numérique peut être une solution. Tatouer, c'est cacher une information dans un document en le détériorant raisonnablement sans augmenter la taille du document et pouvoir récupérer les données dissimulées.

D'autres applications intéressent l'entreprise autour du tatouage numérique des formes. Il s'agit du contrôle d'intégrité servant à vérifier que le maillage n'a pas été modifié (ou que la modification est assez infime pour ne pas avoir d'incidence) ; ou encore de l'enrichissement par des méta-données. Nous verrons que dans notre cas nous souhaitons un tatouage robuste, c'est-à-dire qui résiste à un ensemble de déformations.

Cependant, quelles que soient les informations que nous souhaitons dissimuler, et quelles que soient les applications, il faut une stratégie qui permette de cacher les données intelligemment, de savoir les retrouver et les réordonner. C'est le problème de la synchronisation que l'on retrouve à la fois dans le processus d'insertion des données et dans le processus d'extraction. C'est donc une phase primordiale à la réussite d'une application de tatouage numérique, et c'est d'autant plus vrai concernant le tatouage sur des maillages 3D.

Organisation du manuscrit

Dans ce manuscrit nous exposons nos travaux sur une étape importante du processus de tatouage numérique : la synchronisation.

Tout d'abord dans une première partie nous ferons un rappel détaillé sur les objets que nous manipulons (Chapitre 1) : que ce soit les différentes modélisations possibles d'un objet 3D ou les opérations qui peuvent être appliquées dessus. Dans les chapitres suivants, nous ferons un état de l'art sur les méthodes de synchronisation (Chapitre 2) accompagné des définitions, des différentes applications et des méthodes d'évaluation des algorithmes de tatouage numérique. Puis nous changerons de domaine (Chapitre 3), pour parler des graphes et plus particulièrement des arbres couvrants de poids minimum (ACPM) qui sera la structure de base de nos propositions.

Dans la seconde partie, nous présenterons nos contributions. La première (Chapitre 4) est une étude de la sensibilité des arbres couvrants de poids minimum et de la recherche de critère de mobilité des points au sein de cette structure sans changer les

connexions établies. La seconde (Chapitre 5) est l'utilisation et la validation de ce critère pour un modèle de synchronisation plus robuste.

Nous conclurons en troisième partie, par les prémices d'une nouvelle proposition ainsi que des perspectives de travail éventuelles.

Première partie

Etat de l'art

Chapitre 1

Géométrie et modélisation 3D

Préambule

La 3D est omniprésente dans notre quotidien. A la télévision (dessins animés, publicité, générique), au cinéma (films d'animation, effets spéciaux), dans l'industrie (CAO) ; difficile de trouver un domaine où la 3D numérique n'apparaît pas.

Depuis la fin des années 90 et le début des années 2000 ; la 3D connaît une forte croissance et devient même un argument de vente notamment dans les jeux vidéos et les films d'animation (Toy Story, Pixar Animation Studios, 1995).

Alors que la tendance était plutôt à la Haute Définition (HD), lors des années 2010 on trouve encore le terme de 3D sous une nouvelle approche. On parle de 3D stéréoscopique (Playstation 3 de Sony, salles immersives iMAX 3D) et de 3D auto-stéréoscopique (3DS de Nintendo et concepts de téléphone mobile) avec une nouvelle gamme technologique qui fait son apparition.

Sommaire

1.1 Introduction

Lorsque nous parlons d'un objet 3D, la plupart des gens s'en font une image assez juste du fait de la présence de la 3D au quotidien. Petit à petit, et depuis les années 80, la notion d'objet numérique 3D est entrée dans la connaissance collective par le biais des jeux vidéo, du cinéma et de la publicité.

De manière scientifique, il s'agit d'un ensemble de points représentés numériquement dans un espace à trois dimensions. Avoir une idée précise de ce que peut être un objet 3D est toutefois bien plus compliqué. Alors que le mathématicien manipule sans souci des éléments continus, et donc des éléments de taille infinie ; l'informaticien se limite à l'affichage et au traitement de données appartenant à un ensemble fini.

Bien qu'une image prise par un appareil photo numérique puisse être de plusieurs millions de pixels, les objets numériques de plusieurs millions, voire plusieurs milliards de points restent des objets de taille finie limitée par l'espace mémoire de la machine donnant l'illusion du continu.

Dans ce chapitre, nous nous attardons dans un premier temps sur les modélisations possibles d'un objet en trois dimensions en section 1.2, puis les différentes façons d'organiser ces informations en section 1.3. Ensuite, nous nous attarderons sur les diverses opérations que l'on peut faire sur un document de ce type ; que ce soit des opérations sur le fichier lui-même (1.4), sur la géométrie (1.5) et sur la topologie de l'objet (1.6).

1.2 Du continu vers le discret

Dans le monde de la photographie (2D), le souci de représenter au mieux une scène du monde continu indénombrable vers le monde du numérique dénombrable est une problématique récurrente. Nous ne détaillerons pas cette problématique matérielle qui reste valable dans le cas d'une représentation 3D.

Cependant la représentation d'une image 2D est beaucoup plus simple qu'une représentation 3D. En couleurs ou en niveaux de gris, compressée ou non, une image 2D reste une grille où à chaque élément de l'image (pixel) est associé une valeur numérique correspondant à une intensité lumineuse.

Pour en revenir à ce qui nous intéresse tout le long de ce manuscrit, la représentation 3D, elle est beaucoup plus complexe. Avoir une représentation fidèle, et la plus proche possible de la réalité est tout aussi important. Outre le rendu graphique et la conservation des données, la numérisation est utilisée pour effectuer des opérations de mesure, de comparaison, de découpe, d'assemblage sur des pièces qui peuvent être uniques ou difficiles à manipuler.

Avant de répertorier les modélisations possibles, précisons quelques notions basiques. De manière générale nous notons M un objet 3D, la représentation d'un objet du monde continu associé à une modélisation qui est précisée. Ce que nous appellerons la géométrie de l'objet correspond à un échantillonnage des points de l'espace réel. Cet ensemble est fini et nous notons $v_M \in V_M$ avec $|V_M| = n$, le nombre de points échantillonnés. Les notions nécessaires seront décrites progressivement au cours du manuscrit et selon les besoins.

Nous répertorions, et nous présentons par la suite, plusieurs catégories de modélisations possibles dans le domaine discret : les modélisations surfaciques (1.2.2), les modélisations volumiques (1.2.3) et les modélisations non-structurées (1.2.4). Nous les présentons dans cet ordre du plus "continu" vers le plus "discret".

1.2.1 Formes continues

Les mathématiques offrent de multiples outils servant à décrire des volumes, des surfaces et des courbes. Nous allons passer de manière succincte en revue quelques outils.

Iso-surface. La méthode la plus simple consiste à décrire l'objet 3D par une fonction $f(x, y, z)$ à valeurs dans l'espace des nombres réels. Prenons pour exemple le cas d'une sphère. Toutes les sphères de l'espace peuvent s'écrire sous la forme suivante avec $(x_C, y_C, z_C) \in \mathbb{R}^3$ les coordonnées du centre de la sphère :

$$\begin{array}{rcl} f_C : \mathbb{R}^3 & \longrightarrow & \mathbb{R} \\ \forall (x, y, z) \in \mathbb{R}^3 & \longmapsto & (x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2 + K \end{array}$$

A partir de la formule générale, on peut déterminer une iso-surface, une sphère parmi l'ensemble des sphères possibles. Avec le même exemple, la sphère S de centre C et de rayon r est l'iso-surface définie par l'équation suivante :

$$S_C = \{(x, y, z) \in \mathbb{R}^3 : (x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2 - r^2 = 0\}$$

A partir de la même fonction on peut déduire simplement le volume de la sphère S de centre C et de rayon r, illustrée figure 1.1 :

$$\mathcal{S}_C = \{(x, y, z) \in \mathbb{R}^3 : (x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2 - r^2 < 0\}$$

Dans ce cas le passage de la surface au volume est évident, nous avons à faire à une surface fermée simple. Dans d'autres cas, la notion d'intérieur ou d'extérieur d'une surface peut être impossible à définir. Nous ne prendrons pas en compte ce type de cas.

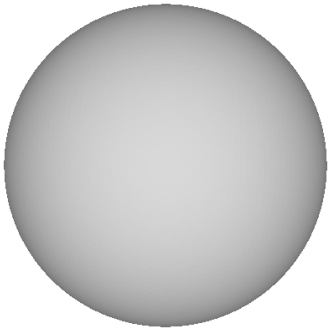


FIGURE 1.1 – L'iso-surface de la sphère : $\{(x, y, z) \in \mathbb{R}^3 : (x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2 - r^2 = 0\}$.

Pour en revenir à quelque chose de plus général, une surface est décrite par une équation (E) avec $f : \mathbb{R}^3 \longrightarrow \mathbb{R}$ telle que :

$$(E)\{(x, y, z) \in \mathbb{R}^3 : f(x, y, z) = 0\}$$

Constructive Solid Geometry. En partant du principe qu'une surface peut être la résultante d'opérations ensemblistes de volumes simples ; ces volumes simples sont appelés primitives et sont définis soit comme des iso-surfaces, soit par des équations paramétriques. L'ensemble des opérations est décrit par un arbre, appelé arbre CSG [Kirsch et Döllner, 2005]. Nous illustrons sur la figure 1.2, un objet construit par une succession d'opérations ensemblistes.

Ce type de modélisation peut être utilisé en CAO, pour des pièces qui seront usinées. Ainsi les opérations de perçage, de filetage, de chanfrenage, etc ... sont faciles à décrire.

NURBS. D'autres outils sont disponibles pour représenter une surface : les NURBS (Non Uniform Rational Basic Spline) qui sont la généralisation des courbes de Béziers, les surfaces implicites, etc. Pour ces dernières approches, les informations sont dans le domaine du continu.

1.2.2 Maillage surfacique

A partir de ce stade nous passons dans le monde discret, le monde de la représentation informatique. Le maillage surfacique (figure 1.3) est la modélisation la plus utilisée notamment pour l'affichage de données visuelles 3D. Autant la machine peut manipuler des données mathématiques comme les représentations citées précédemment, mais elle ne peut pas afficher d'informations de taille infinie. C'est pour cette principale raison que cette modélisation est largement utilisée.

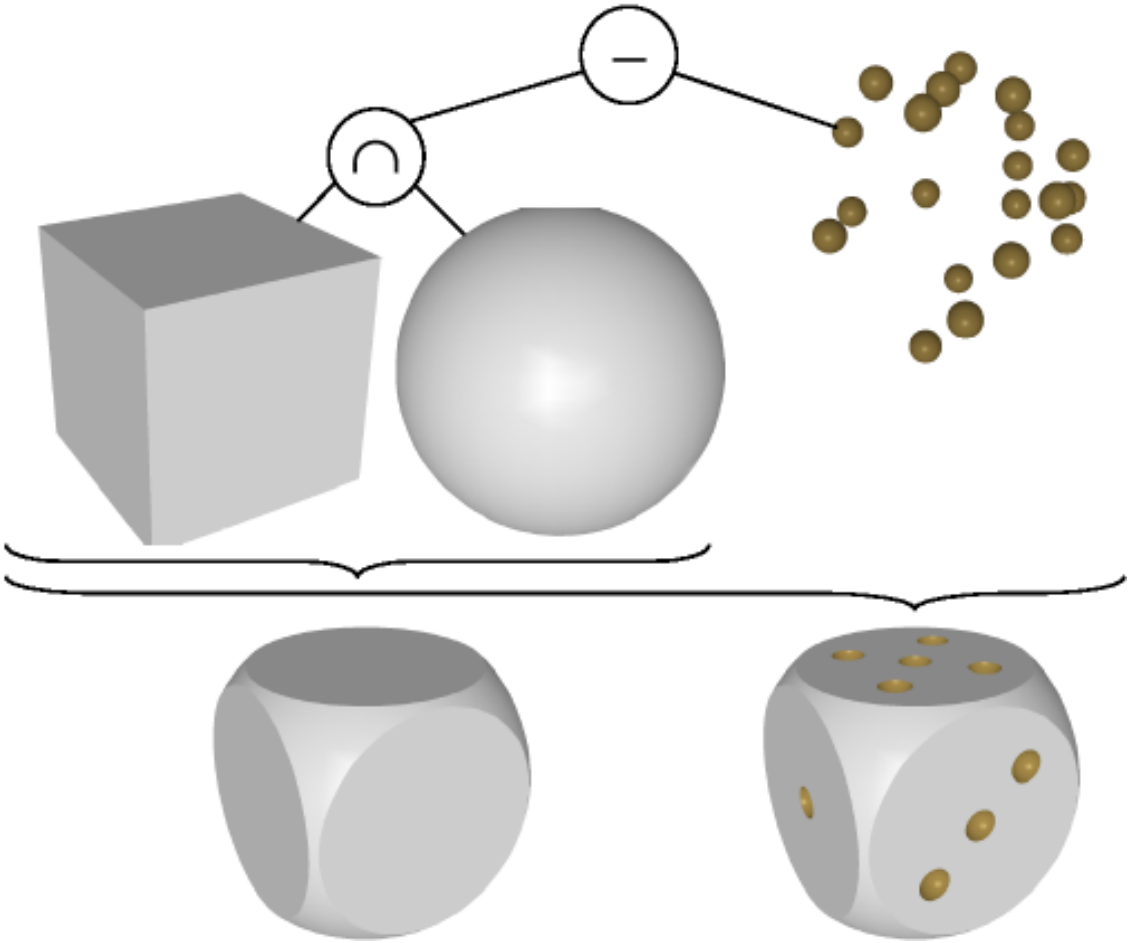


FIGURE 1.2 – Un exemple d'objet construit par CSG. Il s'agit de l'union d'un cube et d'une sphère qui va former un dé à six faces. Puis, pour avoir les points sur le dé, on enlève un ensemble de sphères.

Un maillage surfacique est un ensemble fini de points que nous notons V_M correspondant à la géométrie de l'objet et un ensemble fini de facettes F_M correspondant à la topologie. Dans la majeure partie des cas, les facettes sont des triangles et par conséquent sont des figures planes. Dans des cas particuliers, il se peut que ces facettes ne soient pas planes et peuvent être des figures géométriques dont le nombre de cotés n'est pas constant (dans ce cas nous parlons de maillages mixtes).

Quel que soit le maillage, une facette peut être transformée sans modification de l'ensemble de points en ensemble de triangles. Ainsi, un maillage composé uniquement de triangles est appelé maillage triangulé.

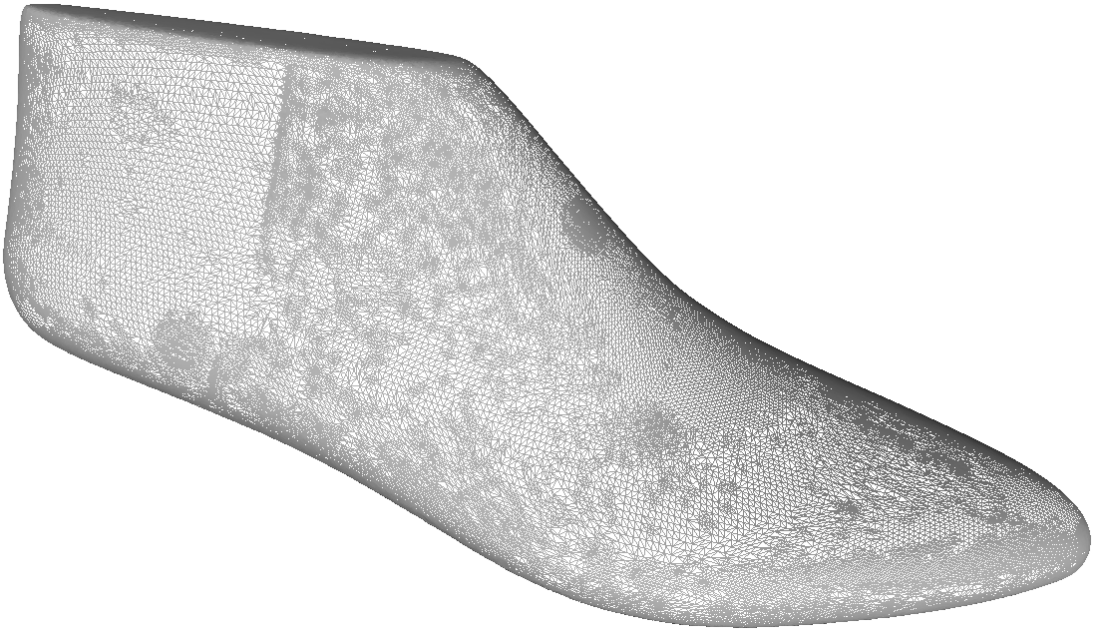


FIGURE 1.3 – Un exemple de maillage surfacique (source : STRATEGIES S.A).

1.2.3 Modélisation volumique

En imagerie 2D, l'unité élémentaire qui compose l'image est le pixel. L'équivalent 3D existe, il s'agit du voxel (*volume element*). Dans la grille régulière 3D, la représentation voxel (figure 1.4) consiste à représenter un cube s'il y a une présence de volume. Du fait que l'unité de volume du voxel est unitaire, il faut augmenter la résolution afin d'avoir une modélisation plus précise mais qui sera beaucoup plus importante en espace de stockage.

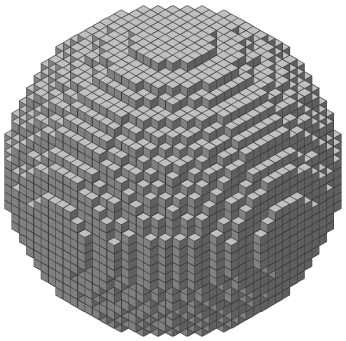


FIGURE 1.4 – Une sphère en voxel (réalisée avec Plotz Modeller).

Pour pallier ce défaut, la représentation *octree*, consiste à représenter le volume avec des voxels de différentes tailles. Ainsi le volume grossier de l'objet est construit à partir d'éléments unitaires, tandis que les détails du volume sont affinés par des éléments de taille plus petite. A partir de cette décomposition, on peut obtenir un objet à différentes résolutions.

1.2.4 Modélisation non-structurée

Ce que nous appelons modélisation non-structurée est une modélisation qui ne donne aucune information sur les relations entre les éléments unitaires.

Soupe de polygones. Les représentations en soupe de polygones (figure 1.5) sont généralement utilisées pour avoir une impression rapide de la géométrie locale d'un objet. Les éléments unitaires sont des polygones. Il n'y a aucune contrainte sur le nombre de cotés que peut compter chaque polygone. Il s'agit par définition d'un ensemble de points planaires. Ces polygones ne sont pas reliés entre eux, c'est pourquoi on parle de soupe de polygones. Ainsi les informations sur la topologie de l'objet restent assez limitées.

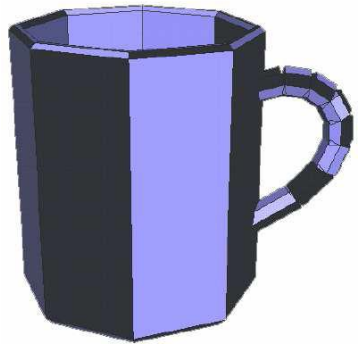


FIGURE 1.5 – Soupe de polygones.

Surfel. Dans ce cas nous sortons un peu du cadre d'application des objets que nous étudions dans le manuscrit. Ici l'objet dispose d'une texture et il est éclairé par un système de sources lumineuses. Les éléments de surfaces sont des disques texturés (figure 1.6) appelés surfel (*surface element*). Connaissant la géométrie de l'objet, la représentation surfel est utilisée pour avoir une idée du rendu visuel de l'objet.

Nuages de points. Le nuage de points (figure 1.7) est la représentation la plus basique d'un objet 3D pour décrire sa géométrie. L'élément unitaire est le point v qui se caractérise par ses coordonnées $v = (v^i)_{(1 \leq i \leq k)}$ dans un espace de dimension k. Dans

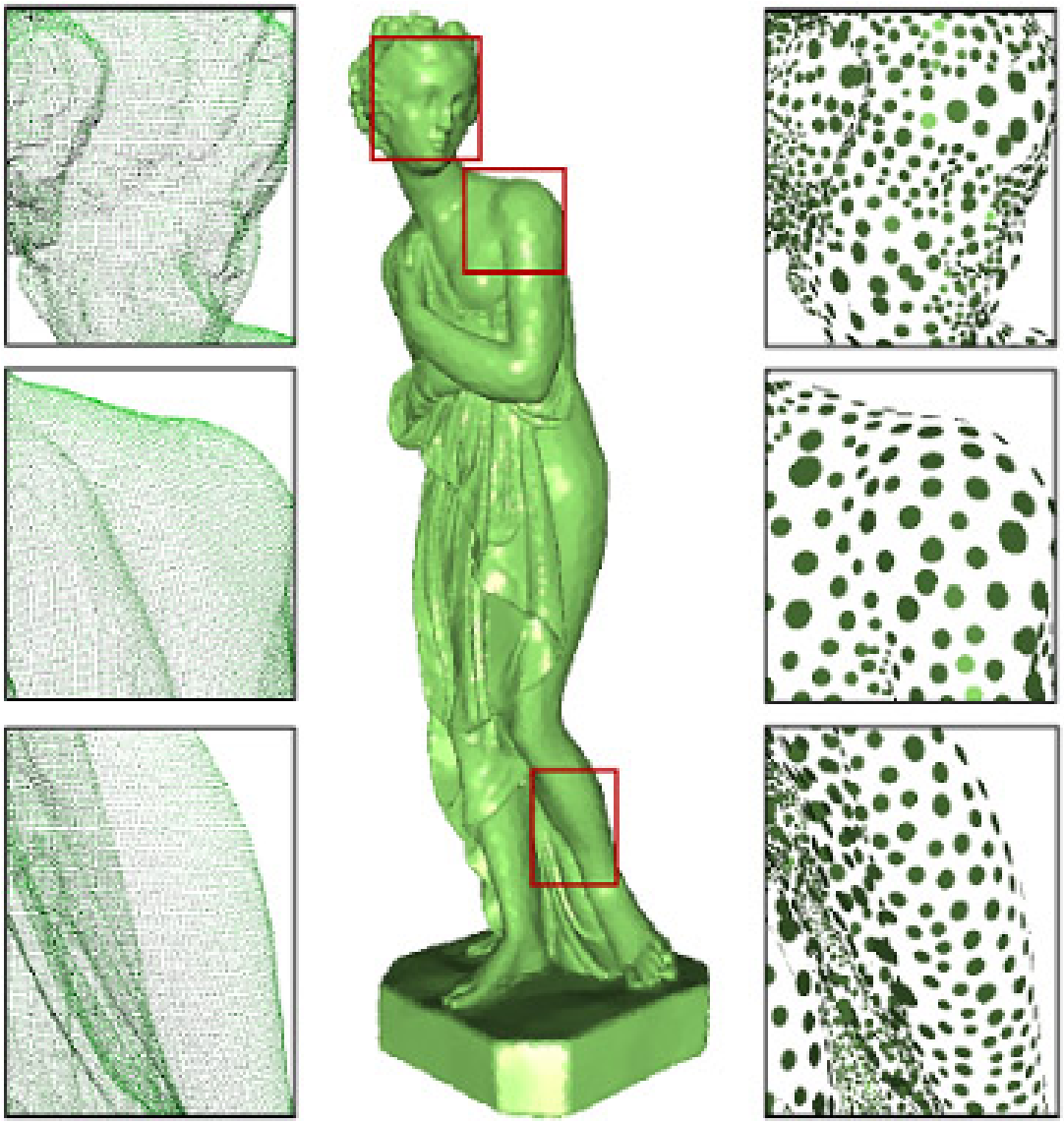


FIGURE 1.6 – Au centre le modèle 3D, sur la gauche des représentations en nuage de points des parties encadrées en rouge et sur la droite une correspondance en surfel (réalisée par Filip Van Bouwel : http ://www.filipvanbouwel.be/master_thesis.php).

le cas le plus commun, nous utilisons le système de coordonnées cartésiennes dans $\mathbb{R}^3$. L'avantage d'une telle représentation est sa simplicité, la liste des points décrit la géométrie globale de l'objet.

De manière formelle et pour la suite du manuscrit nous notons $V_M = \{v_i\}_{(1 \leq i \leq n)}$ l'ensemble des points appartenant au nuage et n le nombre de points dans le nuage.

Cette modélisation peut être issue d'une grande variété de systèmes d'acquisition : scanner, digitaliseur, laser point, laser plan, etc. Ce type de donnée est brut puis généralement traité pour construire des objets avec une modélisation surfacique ou volumique.

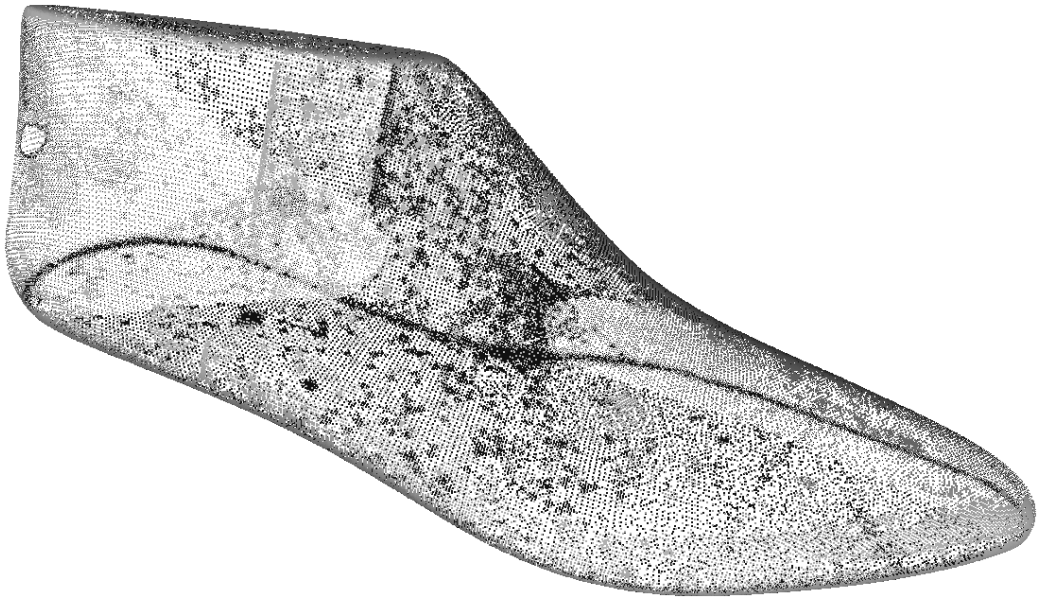


FIGURE 1.7 – Un exemple de nuage de points, correspondant au maillage surfacique représenté à la figure 1.3.

Dans certaines applications, telle que la représentation d'environnement complexe, nous constatons récemment la préférence à une représentation en nuage de points de très grande densité à une représentation de type maillage surfacique, pour des raisons de performance sur des données importantes.

1.2.5 Conclusion

L'objectif de cette section était de montrer la diversité dans les modélisations numériques d'un objet 3D. En ce qui nous concerne, nous travaillerons avec les modèles utilisés dans le cadre de nos applications industrielles. Les fichiers manipulés sont au format STL, obtenus soit après scan 3D, soit auprès des clients. Il s'agit d'un maillage

surfacique triangulé orienté de manière cohérente. Ainsi avec le maillage surfacique, on peut également sans trop de problème travailler également à partir du nuage de points.

1.3 Organisation de l'information

Nous avons vu précédemment qu'un objet 3D pouvait être modélisé de différentes façons en informatique pour différents types d'usage. En ce qui nous concerne, nous travaillons sur des maillages surfaciques sur lesquels il est facile de passer sur les nuages de points.

Cependant, l'organisation de données dans un fichier est également un problème pour le même type de modélisation. Le format que nous utilisons est imposé par l'entreprise, il s'agit du format STL (1.3.3) que nous présentons par la suite.

Afin d'être complet sur la question, nous présenterons en comparaison les formats OBJ (1.3.1), PLY (1.3.2) qui sont des formats d'écriture plus simples et X3D (1.3.4) qui est le format vers lequel l'entreprise se tourne à présent. Nous pourrons discuter des avantages à privilégier un format plutôt qu'un autre.

Dans la prochaine section, nous pourrons ainsi analyser les modifications possibles dans un fichier (sans changer les informations sur la géométrie et la topologie de l'objet) ainsi que les contraintes du passage d'un format à un autre.

1.3.1 Format OBJ

La structure du fichier est très simple, la liste des sommets est énumérée sans ordre défini puis la liste des triangles composant le maillage surfacique triangulé. La syntaxe est elle-même très simple, chaque ligne correspond à un élément (un sommet ou une facette). Pour un sommet il suffit d'écrire la lettre « v » suivie de la valeur de ses coordonnées cartésiennes écrite en flottant et d'un retour à la ligne pour passer au sommet suivant ou commencer la liste des facettes.

Exemple.

```
v -0.0774661 0.459745 -0.0962393
```

Le sommet décrit a pour coordonnées $(-0.0774661; 0.459745; -0.0962393)$.

Pour une facette, la ligne commence par la lettre « f » suivie des numéros de lignes correspondant aux sommets qui la composent. L'ordre des sommets importe peu, traditionnellement ils sont écrits dans le sens direct lorsque l'espace est orienté. Ainsi lors d'un calcul de normale, elle est automatiquement dirigée vers l'extérieur de l'objet.

Exemple.

```
f 7 8 9
```

La facette est composée des sommets écrits aux lignes 7, 8 et 9 du fichier.

Ce type de fichier est très simple et intuitif. Cependant lors du chargement du fichier dans un programme, on ne sait pas à l'avance quelle est la taille de l'objet qui va être traité. Par ailleurs les informations sur la texture de l'objet ne peuvent pas être ajoutées. Dans notre cas ceci importe peu.

1.3.2 Format PLY

Difficile de faire plus simple que le format OBJ, mais le format PLY apporte des fonctions supplémentaires qui comblent en partie ses défauts. Par la présence d'un entête, il est par exemple possible de connaître la taille de l'objet (nombre de sommets, nombre de facettes). Cet entête décrit également la structure globale du fichier.

Par exemple, dans l'exemple qui suit, on apprend que l'objet est constitué de 35 947 sommets et 69 451 facettes. Les facettes sont décrites par une liste, au format *uchar int*, et que cette liste correspond aux indices des sommets. De plus, les sommets sont décrits par 5 caractéristiques : x, y, z, confidence, intensity qui sont des nombres enregistrés dans le fichier au format *float*.

Exemple.

```
ply
format ascii 1.0
comment zipper output
element vertex 35947
property float x
property float y
property float z
property float confidence
property float intensity
element face 69451
property list uchar int vertex indices
endheader
```

En suivant les instructions décrites dans l'entête, on connaît la structure du document. Voici des exemples de sommet et de facette :

Exemple de sommet.

```
-0.0774661 0.459745 -0.0962393
```

Comme on peut le voir, il y a peu de différence avec le format OBJ. Ici il est inutile d'indiquer si nous avons à faire à un sommet ou une facette, vu que l'entête du document indique clairement sa structure.

Exemple de facette.

```
3 7 8 9
```

Dans le cas de la facette, le nombre de sommets qui la compose est indiqué, suivi des indices des sommets.

1.3.3 Format STL

Nous avons vu les formats OBJ et PLY qui listent les sommets et les facettes sans réelle organisation dans le document. Contrairement à ces formats, le format STL, utilisé par l'entreprise, considère qu'une scène 3D est composée d'objets, ces objets sont composés de facettes, qui elles-mêmes sont composées de sommets. Nous avons une décomposition hiérarchique de l'objet, et l'organisation ressemble à ce que pourrait être un document XML. C'est dans la philosophie de XML que ce format a vu le jour, visant à simplifier les échanges de données.

L'autre avantage de cette structure est qu'il est possible de différencier de multiples objets dans la même scène alors que dans les cas précédents, il importe peu qu'il y ait un ou plusieurs objets. Nous concernant, cet avantage est peu important puisque en règle générale nous traitons qu'un seul objet à la fois.

La syntaxe est la suivante. Pour déclarer le début d'un objet dans la scène *solid* suivi du nom que nous souhaitons attribuer à l'objet. Tout ce qui est inscrit entre *solid* et *endsolid* appartiendra à l'objet en question, puis pour déclarer une facette *facet normal* x y x avec x, y et z les valeurs du vecteur normal de la facette qui par convention est orienté vers l'extérieur du volume. Dans la facette, les sommets sont décrits par *vertex* x y z avec x, y et z les valeurs des coordonnées cartésiennes du sommet.

Structure d'un fichier STL.

```
solid monObjet
facet normal x y z
outer loop
vertex x y z
vertex x y z
```

```
vertex x y z
endloop
endfacet
facet normal x y z
outer loop
vertex x y z
vertex x y z
vertex x y z
endloop
endfacet
...
endsolid
```

Remarquons qu'un sommet peut appartenir à plusieurs facettes. Avec cette description de l'objet, il y a une forte redondance de l'information géométrique ce qui pose différents problèmes. Tout d'abord, l'espace mémoire lors du stockage de l'information et l'édition du fichier lors du déplacement d'un sommet par exemple.

1.3.4 Format X3D

Par extension au format STL est venu le format VRML puis son successeur le format X3D. Ce format s'inscrit toujours dans la philosophie du format XML, de syntaxe similaire et facilitant les échanges de données. L'objectif de ce type de document est la représentation d'univers interactifs en 3D. Ce type de format aurait pu s'imposer, avec l'idée d'un web 3D qui a germé au début des années 2000.

Exemple de fichier X3D.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D [...] "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D [...] >
 <Scene>
  <Shape>
   <IndexedFaceSet coordIndex="[...]" solid="false">
    <Coordinate point="[...]"/>
   </IndexedFaceSet>
  </Shape>
 </Scene>
</X3D>
```

Dans la balise *IndexedFaceSet* on retrouve la liste des facettes telles qu'elle pourrait être écrite au format PLY, les indices des sommets composant le triangle suivi de -1 ser-

vant de séparateur. Dans la balise *Coordinate*, on retrouve les coordonnées cartésiennes de chaque point écrit les uns à la suite des autres.

On retrouve les inconvénients des différents formats (OBJ, PLY), sans la redondance que générait le STL, tout en gardant la philosophie d'échange de document au format XML et adapté pour le web.

1.4 Opérations sur les données

Dans cette section, nous allons voir les opérations simples que l'on peut effectuer sur un objet 3D. Nous répertorions ces opérations en trois catégories : les opérations sur les fichiers qui ne changent pas la géométrie ni la topologie de l'objet, les opérations géométriques et les opérations topologiques.

1.4.1 Opérations sur les fichiers

Nous prendrons en compte ici uniquement les opérations sur les fichiers qui n'entraînent aucune modification de la géométrie et de la topologie de l'objet. Nous considérons deux types d'opérations sur les fichiers, la modification interne d'un fichier et le changement de format. Chaque opération sera illustrée par un exemple dans un format, et puisque nous utilisons surtout le format STL, nous essayerons de proposer des modifications en priorité sur ce type de fichier.

1.4.2 Modification d'un fichier

La principale opération sur les fichiers sans changer la géométrie et la topologie est la permutation d'informations. Cela peut être une permutation dans la liste des sommets, la liste des facettes ou encore la liste des sommets décrite dans la facette. Voyons ce que cela donne par un exemple pour chacune de ces modifications :

Permutations des sommets. Étant donné qu'il n'y a pas de liste de sommets dans un fichier STL, nous prenons donc un fichier au format OBJ pour exemple.

```
...
v -0.0061435 0.066575 0.0340161
v -0.0048935 0.066575 0.0340189
v -0.0068935 0.066825 0.0339846
...

...
v -0.0061435 0.066575 0.0340161
```

```
v -0.0068935 0.066825 0.0339846
v -0.0048935 0.066575 0.0340189
...
```

La position des sommets a été modifiée, et donc leur indice a changé. Cette modification entraîne également un changement d'indice dans les facette où se trouvent les sommets pour éviter toute perturbation géométrique.

Permutations des facettes. Toujours en OBJ, même si cette modification peut être faite sur d'autres formats. Nous permutons deux facettes représentées en italique. On remarque que la permutation peut également se faire sur plusieurs lignes, il n'y a pas d'obligation à ce que les lignes se succèdent. Dans le cas de permutation de facettes, ceci n'a aucune incidence sur les sommets.

```
...
f 30 13 29
f 30 29 44
f 31 24 30
...
```

```
...
f 31 24 30
f 30 13 29
f 30 29 44
...
```

Permutations des sommets dans les facettes. Enfin la permutation de sommets dans les facettes, illustrée ci-dessous, qui peut affecter la visualisation de l'objet si le calcul de la norme est fonction de l'ordre des sommets dans la facette. Si la permutation est cyclique, elle respecte l'orientation de l'espace et l'opération n'est pas visible.

```
...
f 31 24 30
...
```

```
...
f 30 31 24
...
```

1.4.3 Changement de format

Le changement de format est une opération sur les fichiers qui ne doit pas dégrader l'objet. Parmi les formats présentés le passage d'un format à l'autre ne pose aucun problème. Les informations communes à tous sont les coordonnées des sommets dans l'objet et sont dans chaque format enregistré en tant que des nombres décimaux avec une précision pouvant aller jusqu'à 10^{-7}.

Concernant les informations supplémentaires telles que les valeurs des coordonnées du vecteur normal dans une facette (pour le format STL), la convention sur l'ordre des index des sommets qui composent une facette, elles peuvent être calculées à la volée et ne posent aucun problème lors du changement de format.

1.5 Opérations géométriques

Contrairement aux opérations sur les fichiers, les opérations géométriques modifient la position des points. Elles peuvent conserver les rapports de distances, c'est ce que nous allons voir avec toutes les isométries ; mais aussi déplacer le point pseudo-aléatoirement par bruitage, ou encore déplacer le point en fonction de la position de ses voisins par lissage, ou encore supprimer des points par quantification.

1.5.1 Isométries

Une isométrie est une opération géométrique qui déplace les points de l'objet tout en conservant un rapport de distance constant entre les points. D'un point de vue mathématique, une isométrie est une application affine d'un espace affine dans un autre. Dans notre cas l'espace affine de départ et d'arrivée est le même, il s'agit de l'espace vectoriel $\mathbb{R}^3$ auquel on associe une mesure, la distance euclidienne $l_2(.)$ par exemple.

A partir de cette définition, décrire une isométrie est relativement simple. Soit $\mathcal{I}$ une isométrie :

$$\mathcal{I} : X \longrightarrow A \cdot X + B$$

X est un point de $\mathbb{R}^3$, $A \in \mathcal{M}_{3,3}$ une matrice et $B \in \mathbb{R}^3$ un vecteur. Avec ces notations on retrouve toutes les opérations de transformation bien connues dont les isométries.

— les translations, $A = I$ (la matrice identité) et $B \in \mathbb{R}^3$;

— les homothéties, $A = k \cdot I$ avec $k \in \mathbb{R}^*$;

— les rotations, A une matrice du groupe spécial orthogonal $SO(\mathbb{R}^3)$;

— les isométries affines, avec $\mathcal{I}$ un automorphisme orthogonal ;

— les similitudes.

1.5.2 Bruitage

Le bruitage est un phénomène physique qui peut intervenir lors d'une communication. Cela provoque des erreurs dans le signal qui la plupart du temps peuvent être corrigées automatiquement par des outils adaptés (codes correcteurs d'erreur). Bruiter ses données permet généralement de tester la stabilité de son application. En fonction des résultats que l'on veut obtenir cette phase peut être intéressante.

En imagerie 2D, bruiter une image c'est perturber les valeurs des intensités lumineuses des pixels. Dans notre cas, nous perturberons la position des points sur les trois directions de l'espace.

Pour modéliser un bruit on utilise généralement des outils statistiques. On suppose que le comportement aléatoire du bruit est régi par une loi de distribution. Cette loi peut suivre une loi uniforme, gaussienne, de Poisson, etc.

Grâce aux bonnes propriétés mathématiques des fonctions gaussiennes ; on suppose que le bruit suit une loi normale gaussienne que nous notons $\mathcal{N}(\sigma, \mu)$ où σ est l'écart-type du bruit et μ sa moyenne.

1.5.3 Lissage

La plupart des algorithmes de lissage déplace les sommets du maillage sans modifier les arêtes. L'un des algorithmes le plus simple est le filtre Laplacien [Sorkine, 2006]. C'est un algorithme de complexité linéaire, donc un algorithme rapide. A chaque étape, le principe est de déplacer chaque point au barycentre de ses voisins. Le problème de cet algorithme est que le maillage se rétracte sur lui-même. En augmentant les itérations un grand nombre de fois, les sommets tendent à converger vers le centre de gravité du maillage. Cette approche a été améliorée dans [Taubin, 2000] pour résoudre le problème de rétractation du maillage.

1.5.4 Quantification

La quantification [Cirio *et al.*, 2010] des coordonnées des sommets est aussi une opération de modification de la géométrie du maillage. C'est une méthode très utilisée dans le domaine de compression de maillage avec pertes. Le principe est d'arrondir uniformément chaque coordonnée de chaque sommet $v_i = (x_i, y_i, z_i)$, par exemple en fixant r comme paramètre de quantification [Wang *et al.*, 2010], chaque coordonnée est arrondie à la puissance de 2^r la plus proche.

1.6 Opérations topologiques

L'objectif de la simplification est de réduire le nombre d'éléments qui compose le maillage triangulaire. Plusieurs approches sont possibles.

Une méthode propose de choisir des sommets dans le maillage complexe qui seront supprimés [Schroeder *et al.*, 1992]. Les trous créés sont remaillés en fonction d'un nombre de facettes souhaité tout en respectant au mieux la géométrie de l'objet. Un sommet est candidat à la suppression si et seulement si le remaillage après sa suppression donne un résultat proche au maillage original.

L'avantage de ce genre de technique est que le maillage simplifié est composé d'un sous-ensemble de points du maillage original. La qualité perceptuelle peut être améliorée en déplaçant ces points de manière à réduire les distorsions dans le maillage [Garland et Heckbert, 1997].

D'autre méthodes existent, elles se basent sur des regroupements de points [Rossignac et Borrel, 1993], ou encore passant par les représentations voxel [He *et al.*, 1995] combiné avec l'algorithme *marching cube* [Lorensen et Cline, 1987]. Ces méthodes sont utilisées principalement pour avoir une représentation d'un même objet à différents niveaux de détails. Ces applications sont utilisées en visualisation afin de simplifier les maillages pour avoir une navigation fluide tout en ayant un rendu équivalent aux objets originaux.

De manière plus progressive, au lieu de supprimer des sommets, nous nous basons sur la contraction des arêtes pour simplifier le maillage. Cette contraction se fait par exemple en fonction de la distance qui sépare les points reliés. Plus les points sont proches, plus il y a de chances que l'arête qui les relie soit contractée. Ce seuil est réglé par l'utilisateur.

Cette technique a été généralisée [Hoppe, 1996]. La réduction des arêtes se fait en fonction d'une fonction d'énergie à minimiser. Cette fonction permet à la contraction de s'adapter au plus proche des caractéristiques du maillage, minimisant ainsi la distorsion.

1.7 Conclusion

L'objectif de ce chapitre est de montrer les supports de notre travail. Il est important de voir que derrière le mot "objet 3D" se cachent en fait plusieurs familles de représentation, chacune adaptée à différentes applications.

De plus, on voit que l'organisation des informations peut être aussi totalement différente, plus ou moins simple à créer et à modifier. Pour un même maillage surfacique, il existe plusieurs façons de le coder. Cependant, nous sommes plus familiers avec ce type de diversité.

Enfin, nous avons survolé différentes opérations qui peuvent être effectuées sur les fichiers et sur les maillages surfaciques. Nous verrons ensuite que ces modifications peuvent correspondre à des "attaques" volontaires ou non dans le processus de protection des objets.

Chapitre 2

Insertion de données cachées

Préambule

Les premières techniques de dissimulation d'information datent de l'Antiquité. Dans La vie des XII César écrit par Suétone (70-127), les écrits révèlent une première méthode de cryptographie. Il s'agit du code de César dont le principe est de remplacer chaque lettre par celle qui suit dans l'ordre alphabétique. Pour décrypter le message, il suffit donc de remplacer chaque lettre par celle qui précède.

Une autre méthode de dissimulation d'information consistait à prendre un messager, souvent un esclave, on lui rasait le crâne ; puis on lui tatouait le message qu'il fallait transmettre. Une fois que ses cheveux avaient repoussé, il allait à la destination souhaitée. S'il se faisait arrêter par l'ennemi, le messager ne disposait d'aucun document sur lequel se trouvait le message et de plus il ne connaissait pas le contenu du message qu'il transportait ; rendant ainsi la fouille et l'interrogatoire totalement inefficace aux ennemis.

Par ces deux techniques antiques, nous voyons deux grands axes de la dissimulation d'information. Tout d'abord, la cryptographie qui consiste à rendre le message illisible, et l'insertion de données cachées qui consiste à utiliser un support anodin pour transporter un message indétectable a priori.

Sommaire

2.1 Introduction

Comme nous l'avons évoqué, il existe deux grandes méthodes pour la dissimulation d'information. Soit nous la rendons illisible par cryptographie, soit nous la cachons dans un autre support par des méthodes d'insertion de données cachées. Nous nous intéressons ici à l'insertion de données uniquement.

Avec l'arrivée du numérique, les premières applications d'insertions de données cachées sur du contenu multimédia numérique se sont faites sur des images et du son dans les années 80, puis de la vidéo et depuis une dizaine d'années sur des données 3D. Ce qui fait du tatouage 3D une science jeune.

Dans cette même branche, deux méthodes se différencient. Cela dépend de la fonction du message dans le support. Soit nous voulons dissimuler le message de manière à ce qu'il soit indécelable visuellement et statistiquement, on parle alors de stéganographie. L'exemple évoqué dans le préambule de ce chapitre en est un exemple, et une des applications est donc la communication secrète. Un utilisateur quelconque ne doit pas se douter de la présence d'un message dans le support.

Soit nous cherchons à ce que le message soit intimement lié au support, un tatouage caché et de manière la plus indélébile possible au risque de dégrader la qualité du support. Il y a dans ces cas toujours un compromis à faire entre l'imperceptibilité de la marque insérée et la résistance de la marque sur le support. On parle alors de tatouage numérique, et ces techniques sont utilisées lorsqu'on cherche à protéger un document par exemple. Dans ce chapitre, nous détaillerons le principe du tatouage numérique (2.2) ainsi que quelques applications possibles (2.3). Puis nous entrerons petit à petit dans le cœur des travaux présentés, axés sur une étape du tatouage numérique : la synchronisation (2.4).

2.2 Tatouage numérique

Une application de tatouage numérique se divise en deux algorithmes. Le premier est un algorithme d'insertion. Son rôle est de dissimuler un message m dans un support numérique X pour obtenir un support tatoué Y. Nous illustrons de manière très simplifiée cet algorithme sur la figure 2.1.

Cette insertion peut être totalement indépendante du support. Dans ce cas nous parlons de tatouage non-informé. En imagerie, par exemple, en prenant la méthode qui consiste à changer les bits de poids faible [Chan et Cheng, 2004] pour les remplacer par un bit du message, cette modification ne prend pas en compte la valeur initiale du pixel, ni les valeurs des pixels voisins.

C'est avec la redécouverte des résultats connus en traitement du signal [Costa, 1983] qu'une nouvelle génération de méthode de tatouage sur des images fait son apparition, il s'agit du tatouage informé qui consiste à prendre en compte les caractéristiques du signal pour l'insertion de l'information. Ces nouvelles méthodes sont bien plus performantes en obtenant en général de meilleurs compromis entre la quantité d'information à dissimuler et la détérioration engendrée sur l'image.

Dans le cadre du tatouage sur des objets 3D, dès que nous prenons en compte la géométrie ou la topologie de l'objet, le tatouage est généralement informé.

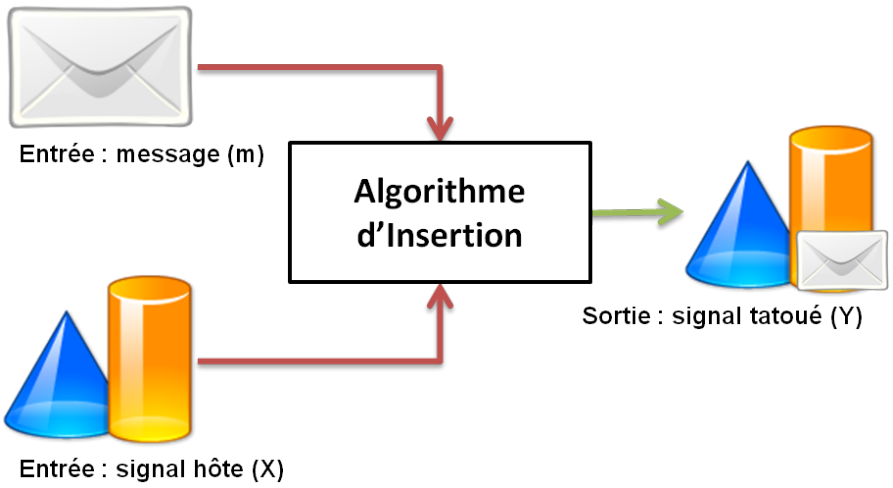


FIGURE 2.1 – Schéma simplifié de l'algorithme d'insertion.

Lors de la réception d'un support tatoué Y', l'algorithme d'extraction a pour objectif de récupérer m', le message dissimulé. Généralement, le support tatoué suffit à la récupération du message, nous parlons d'extraction aveugle. Nous avons illustré cette opération de manière très simplifiée, comme précédemment, sur la figure 2.2. Dans le cas de tatouage non aveugle, le destinataire doit avoir en possession le support original X ayant servi à l'insertion.

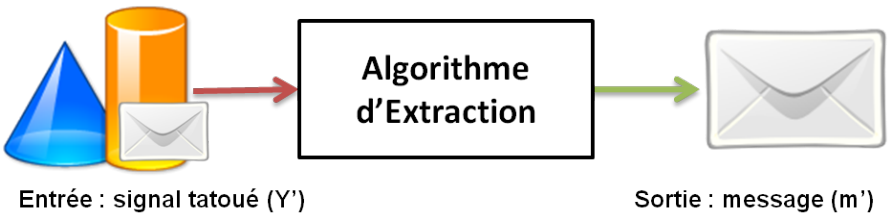


FIGURE 2.2 – Schéma simplifié de l'algorithme d'extraction.

A présent, nous allons détailler ces deux algorithmes dans les sections 2.2.1 et 2.2.2 et nous attarder ensuite sur la phase de synchronisation.

2.2.1 Algorithme d'insertion

L'objectif est de dissimuler un message m dans un support X (figure 2.3). On appelle ce support, un signal hôte. Il peut être un document multimédia (son, image, vidéos, objet 3D), une application, un texte. Tout document numérique peut servir de signal hôte. Dans notre cas, ce sera un maillage surfacique ou un nuage de points.

Avant d'être inséré à proprement parler dans le signal hôte, le message peut subir des transformations ; soit pour insérer un maximum de bits (en faisant une compression par exemple), soit pour hiérarchiser l'information, soit pour augmenter la protection du message (par exemple, en cryptant les données avant l'insertion), etc. Nous allons passer sous silence cette transformation qui n'est pas forcée d'exister. Le principal est d'avoir un mot binaire à dissimuler.

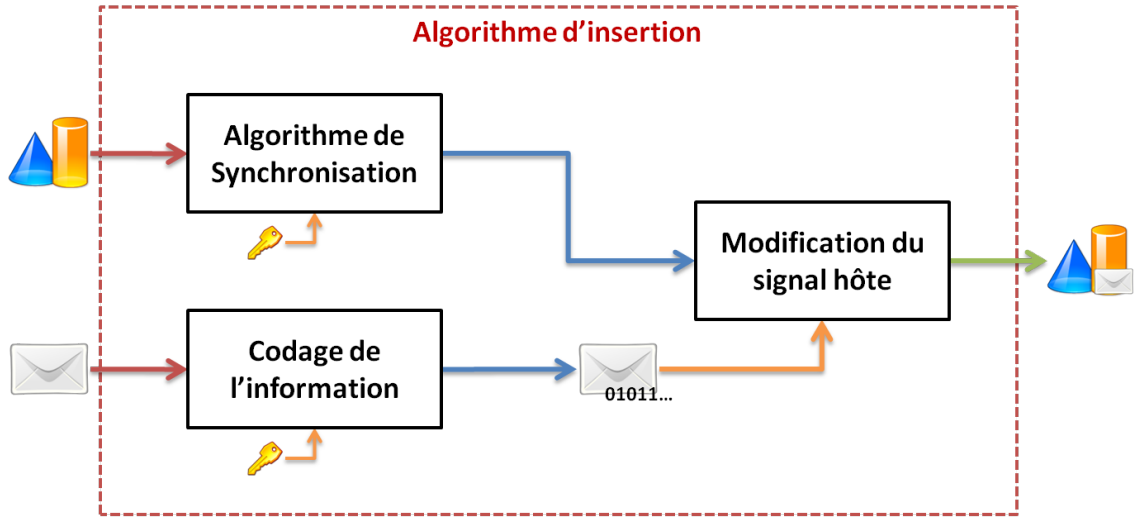


FIGURE 2.3 – Schéma d'insertion d'un message par tatouage

La dissimulation se fait par une modification du signal hôte. C'est une opération qui détériore le signal original, qui est généralement irréversible. On parle de tatouage réversible quand cette opération l'est à l'extraction ; c'est-à-dire qu'on retrouve le signal original lorsque le message est extrait du signal tatoué. En fonction du bit à coder (0 ou 1 ou un mot binaire plus long), la modification du signal est différente. L'ensemble des altérations que nous effectuons sur le signal est appelé la "marque". A chaque message, et à chaque support correspond une marque qui doit être unique. Si l'algorithme a réussi à insérer tout le message dans le signal, l'opération est réussie, on obtient en sortie un signal tatoué Y.

Cependant, ces modifications ne peuvent pas se situer n'importe où dans le signal. Le rôle de la synchronisation, cette étape qui précède l'insertion du message, est de repérer dans le signal les "zones" propices à la dissimulation d'information. Nous y viendrons plus en détail prochainement.

2.2.2 Algorithme d'extraction

Lors de la transmission, le signal Y peut subir des modifications dues à la communication, par des attaques d'utilisateurs malveillants, ou par des modifications volontaires (par exemple, un changement de format). Nous notons Y' le signal tatoué reçu. L'objectif de l'algorithme d'extraction (figure 2.4) est de récupérer le message qui est caché dans notre signal reçu. Dans le cas général, le signal original X n'est pas connu lors de l'extraction ; nous parlons d'extraction aveugle.

Nous avons dit précédemment, que la synchronisation était de repérer les "zones" dans lesquelles un ou plusieurs bits pouvaient être cachés. Ce sont donc dans ces mêmes zones que se trouve l'information. La même méthode est appliquée, c'est pourquoi la synchronisation est une étape clé dans un processus de tatouage ; elle intervient à l'insertion et à l'extraction du message. Une mauvaise synchronisation compromet l'efficacité de la méthode.

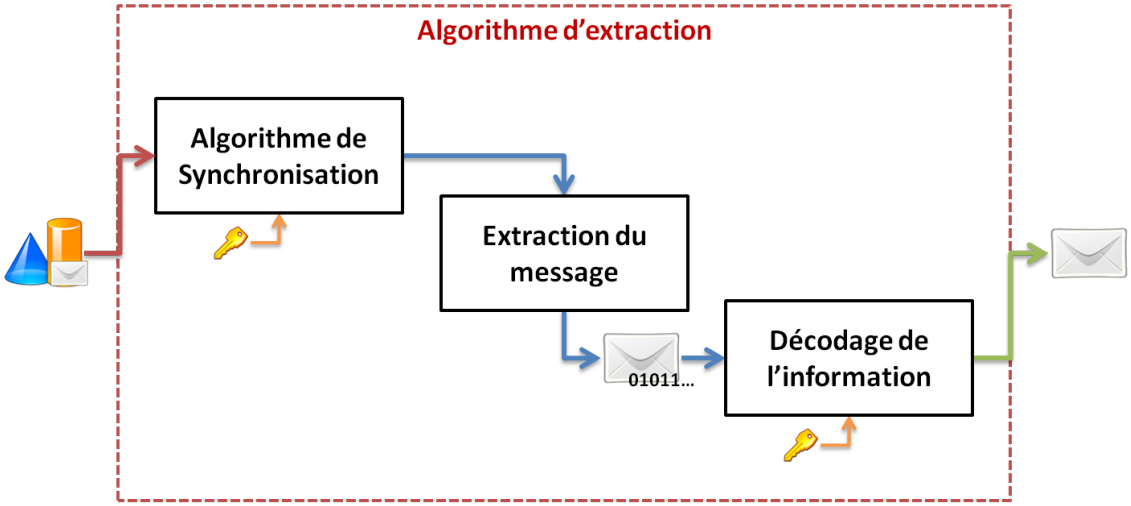


FIGURE 2.4 – Schéma d'extraction d'un message par tatouage.

Une fois synchronisé, on applique la méthode inverse à l'insertion. A moins que la méthode soit réversible, nous ne modifions pas le signal tatoué. Nous nous servons de la synchronisation pour détecter dans les zones tatouées quel bit (ou quel mot binaire) a été dissimulé. Ainsi on recompose le message binaire bit à bit, jusqu'à retrouver un message. Si l'opération générale s'est bien déroulée, on devrait trouver le message envoyé.

2.2.3 Algorithme de synchronisation

La synchronisation, illustrée figure 2.5, est la phase clé du schéma de tatouage. Tout d'abord, elle transforme l'objet 3D pour le placer dans de bonnes conditions initiales. Cela correspond par exemple à un changement de domaine (transformation en ondelettes, transformation dans un domaine fréquentiel, changement de dimension, ...) ou encore dans le domaine spatial à un repositionnement de l'objet (rotation, homothétie, analyse en composantes principales). C'est le genre d'opération qui permet de faire face à toute déformation isométrique.

A l'insertion et à l'extraction, il est important que l'objet se trouve dans les mêmes conditions initiales. Ensuite nous parcourons les données. Dans nos études nous restons dans le domaine spatial, les données que nous avons à parcourir sont toujours des données géométriques. Si le domaine est différent, le parcours peut se faire selon les fréquences par exemple. Qui dit parcours, dit point de départ et sens de parcours. Or, parcourir des données géométriques est également un problème difficile.

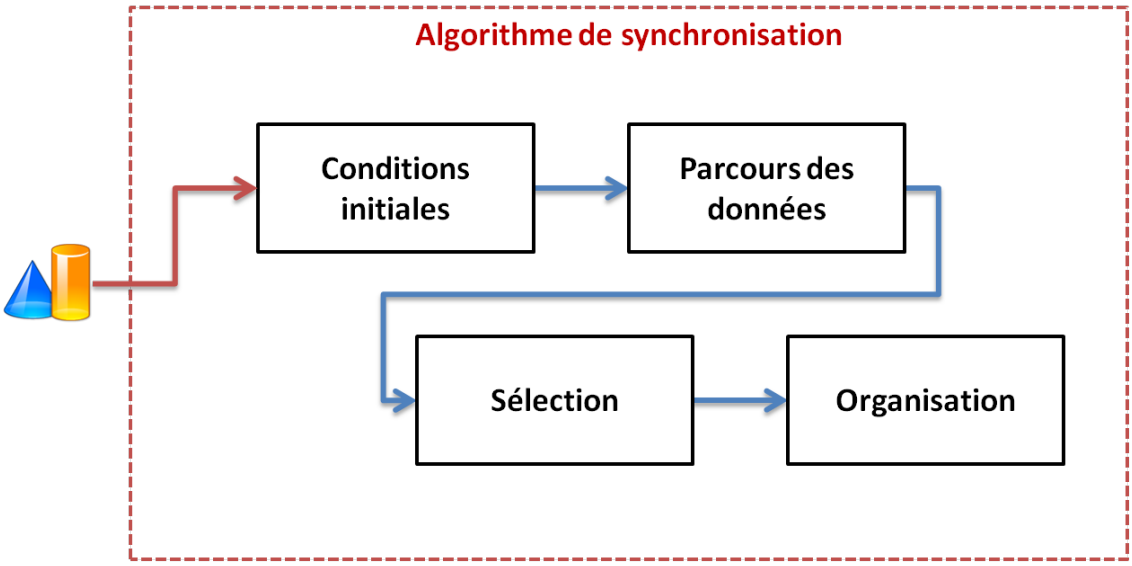


FIGURE 2.5 – Schéma de synchronisation d'un objet.

Lors du parcours, il va falloir sélectionner les "entités" susceptibles d'être propices à l'insertion d'une information. Ces "entités" peuvent être, dans le domaine spatial, des éléments géométriques : sommets, arêtes, facettes, quadrangles, etc. La sélection se fait en fonction de l'algorithme d'insertion et de la fonction du message.

Enfin, une fois les données sélectionnées, il faut les ordonner. Cela peut être en fonction de l'ordre du parcours, ou de manière différente. A l'arrivée, nous avons un ensemble d'entités sélectionnées, ordonnées, prêtes à être modifiées lors de l'étape d'insertion ou détecter les bits cachés lors de l'étape d'extraction.

2.2.4 Critère d'évaluation d'un algorithme

Une fois l'algorithme créé, il convient de pouvoir l'évaluer afin de pouvoir comparer les performances entre les différents algorithmes. Pour les algorithmes de tatouage, nous l'évaluons suivant les trois critères suivants :

— capacité (ou payload) : c'est la quantité d'informations que nous pouvons dissimuler en moyenne dans un signal hôte. Pour une image elle s'exprime en bit/pixel et pour un objet 3D en bit/sommet ou bit/facette ;

— robustesse : elle s'exprime généralement en TEB (Taux d'Erreur Binaire) et évalue la résistance de la marque à différentes attaques avec un jeu de paramètres fixés. Nous saurons si la méthode est fragile, ou robuste à certain type d'attaques :

$$TEB = \frac{\text{nombre de bits erronés}}{\text{nombre de bits total}};$$

— imperceptibilité : en insérant de l'information nous dégradons un peu le signal hôte. Suivant les applications, nous souhaitons que le modèle 3D garde toujours une certaine précision, il faut donc ne pas trop le détériorer. Ceci se mesure par des calculs de distances entre deux supports, avec (ou non) la mise en place d'un modèle psycho-visuel.

Nous pouvons mesurer la distorsion d'un modèle 3D soit de manière objective, soit de manière perceptuelle. De manière objective, prenons la moyenne quadratique de la distance d'une surface à celle que nous voulons comparer, nous obtenons la distance RMS [Armstrong et Collopy, 1992] (Root Mean Square Error). C'est une distance qui compare deux surfaces. Nous pouvons également l'adapter pour comparer deux nuages de points.

Soient deux maillages surfaciques $\mathcal{M}$ et $\mathcal{M}'$ (dans notre contexte, ce maillage est celui qui est tatoué) ; $\mathcal{S}$ et $\mathcal{S}'$ leurs surfaces correspondantes et d une distance de $\mathbb{R}^3$, alors la distance RMS s'exprime de la façon suivante :

$$d_{RMS}(\mathcal{S}, \mathcal{S}') = \sqrt{\frac{1}{S} \int \int_{p \in S} d(p, \mathcal{S}')^2 d\mathcal{S}}.$$

De manière générale, la distance RMS n'est pas symétrique. C'est pourquoi en général nous prenons la distance MRMS :

$$d_{MRMS}(\mathcal{M}, \mathcal{M}') = \max\{d_{RMS}(\mathcal{S}, \mathcal{S}'), d_{RMS}(\mathcal{S}', \mathcal{S})\}$$

Au lieu de prendre la distance moyenne, nous prenons l'écart maximal entre les deux surfaces, nous retrouvons la distance de Hausdorff d_H [Belogay *et al.*, 1997] que nous illustrons figure 2.6.

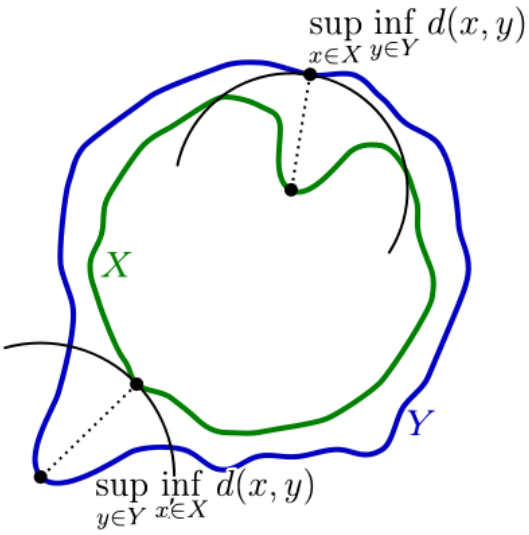


FIGURE 2.6 – Illustration de la distance de Hausdorff (source : Wikimédia).

$$d_H(\mathcal{S}, \mathcal{S}') = \max\{\sup_{x \in S} \inf_{y \in S'} d(x, y), \sup_{y \in S'} \inf_{x \in S} d(x, y)\}$$

Pour une mesure perceptuelle, nous avons par exemple, la distance MSDM [Belogay *et al.*, 1997] (Mesh Structural Distortion Measure) qui prend en compte les fonctions de courbure $L(p, q)$, de contraste $C(p, q)$ et de structure $S(p, q)$, données ci-dessous en notant μ_p la moyenne, σ_p l'écart-type et σ_{pq} la covariance de la courbure sur le maillage local :

$$L(p, q) = \frac{||\mu_p - \mu_q||}{\max(\mu_p, \mu_q)}$$

$$C(p, q) = \frac{||\sigma_p - \sigma_q||}{\max(\sigma_p, \sigma_q)}$$

$$S(p, q) = \frac{||\sigma_p \cdot \sigma_q - \sigma_{pq}||}{\sigma_p \cdot \sigma_q}$$

La valeur de cette distance est comprise entre 0 (lorsque les deux maillages sont parfaitement identiques) et 1 (lorsque les deux maillages sont très différents). Ainsi la distance MSDM s'exprime de la façon suivante :

$$d_{LMSDM}(p, q) = \frac{2}{5} L(p, q)^3 + \frac{2}{5} C(p, q)^3 + \frac{1}{5} S(p, q)^3$$

$$d_{MSDM}(\mathcal{M}, \mathcal{M}') = \sqrt[3]{\frac{1}{n}\sum_{j=1}^{n} d_{LMSDM}(p_j, q_j)^3}$$

Ne pouvant pas maximiser les trois critères, chaque méthode doit faire des compromis et trouver un équilibre entre robustesse, capacité et imperceptibilité. Plus récemment, deux nouveaux critères aussi importants sont ajoutés à l'évaluation des algorithmes. Ils sont illustrés sur la figure 2.7 :

— sécurité : en se plaçant dans les hypothèses du principe de Kerckhoff [Kerckhoffs, 1883], l'utilisateur malveillant dispose des algorithmes de tatouage (insertion et extraction) de la méthode qu'il tente d'attaquer. Le but est de savoir, si nous pouvons avoir une connaissance de la clé qui permet le codage et l'extraction du message ;

— complexité algorithmique : elle permet d'avoir une idée de la vitesse d'exécution de l'algorithme, et de l'espace mémoire nécessaire. Ces informations sont exprimées en fonction de la taille de la donnée, pour un objet 3D nous prenons son nombre de points. De plus amples informations seront décrites dans le chapitre 3.

2.3 Applications du tatouage numérique

En fonction des critères d'évaluation des algorithmes, divers types d'applications sont possibles. Nous présentons ici des applications variées : de communication secrète (2.3.1), d'enrichissement de contenu (2.3.2), de protection multimédia (2.3.3), de traçage de traîtres (2.3.4) et de contrôle d'intégrité (2.3.5).

2.3.1 Stéganographie

Précédemment, nous avons présenté le tatouage numérique par les chaînes de communication. L'utilisation la plus courante est la communication secrète, également appelée stéganographie. L'objectif de la communication secrète (figure 2.8) est de dissimuler dans le signal hôte une information qui est indécelable aussi bien visuellement que statistiquement. La stéganalyse est la science qui étudie la présence ou non d'un message au sein d'un médium.

En fonction des critères d'évaluation, une méthode visant une application dans la stéganographie doit être imperceptible visuellement et statistiquement, mais également à haute capacité, l'objectif étant de transmettre par exemple une image, une vidéo, un texte complet. Pour des raisons pratiques, nous souhaitons que le processus d'insertion et d'extraction soient relativement rapides.

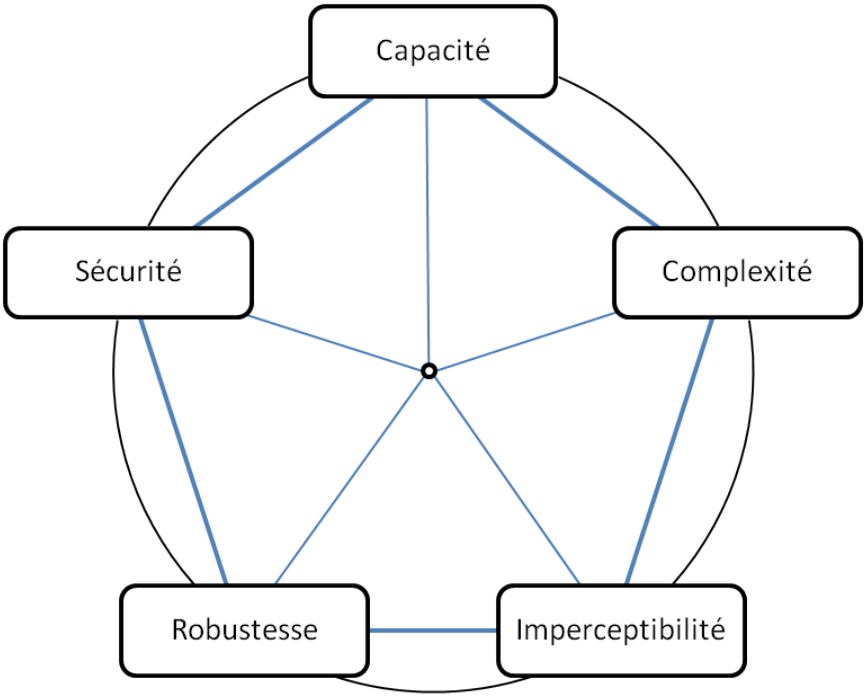


FIGURE 2.7 – Schématisation des compromis pour l'évaluation des algorithmes de tatouage complétée par les critères de sécurité et de complexité.

2.3.2 Enrichissement

Un signal peut être accompagné d'un ensemble de méta-données : nom de l'auteur, année de parution, type de données, références éventuelles, titre, etc. L'enrichissement consiste à prendre ces méta-données textuelles, visuelles, sonores, personnelles et de les dissimuler dans le document référent. Nous pouvons nous demander quelle est l'utilité de ce genre d'application. Ceci se justifie principalement par l'économie de l'espace mémoire (figure 2.9).

Pour rappel, le principe du tatouage est de dissimuler dans un signal une information sans augmenter la taille du signal. Idéalement, nous dissimulons l'ensemble des méta-données dans le signal original sans augmenter sa taille. Nous économisons donc l'espace des méta-données insérées. La méthode de dissimulation doit donc impérativement être de très haute capacité, tout en préservant la qualité du signal hôte.

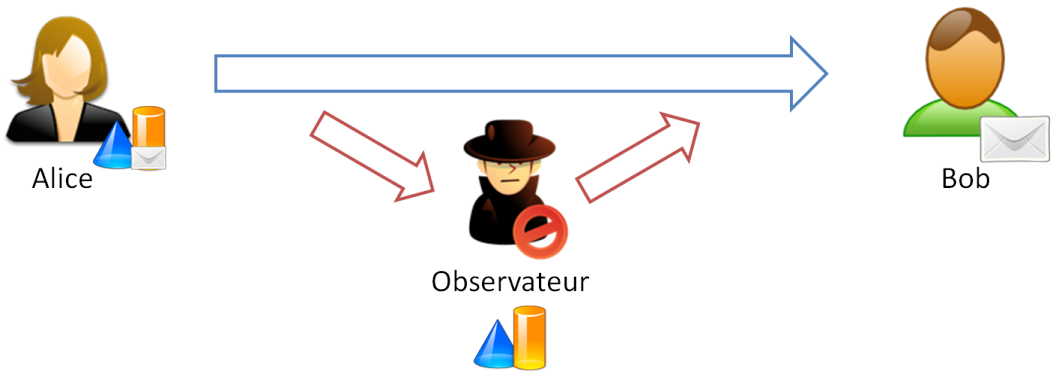


FIGURE 2.8 – Illustration d'un exemple d'application de communication secrète. Imaginons que Alice et Bob sont des prisonniers qui veulent s'échanger des messages. Chaque communication est surveillée par un gardien qui analyse les messages transmis. L'objectif des prisonniers est de planifier un plan d'évasion sans que le garde ne s'en doute. C'est ce principe de communication secrète qui définit la stéganographie qui est appliqué. Dans notre schéma de communication Alice insère un message dans le signal hôte qui est transmis à Bob. Bob pourra extraire puis lire le message inséré. Si un observateur quelconque intercepte le message, alors il ne faut pas qu'il se doute que le signal contient une information supplémentaire.

2.3.3 Protection

Également, dans la protection de données numériques, l'insertion de données cachées peut être une solution (figure 2.10). Dans ce genre d'application, généralement, nous cherchons à dissimuler des informations de propriété des données, droits d'auteurs, ou des informations d'identification des données. Nous voulons que la marque reste dissimulée même si elle subit des attaques par des utilisateurs malveillants de manière à prouver son appartenance ou de permettre de l'identifier correctement.

Pour en revenir aux conditions d'évaluation de l'algorithme, il faut que les processus de synchronisation et d'insertion soient robustes à un ensemble d'attaques qui auront été prises en considération. Ici la capacité d'insertion est moins importante, nous voulons cacher une petite séquence dans le signal, nous souhaiterons donc qu'elle soit multi-bits de faible capacité.

En fonction du cadre de l'application, il se peut qu'il n'y ait aucune contrainte de temps. Si par exemple, pour protéger la donnée et dissimuler la marque il faut une journée de calcul, la méthode peut être acceptée. Clairement, s'il s'agit d'une application utilisateur qui est visée nous imposerons une contrainte de temps raisonnable. La contrainte de complexité ne dépend pas du type de l'application mais de l'usage qui en sera fait.

Pour ce genre d'application, nous souhaitons donc privilégier l'aspect de robustesse.

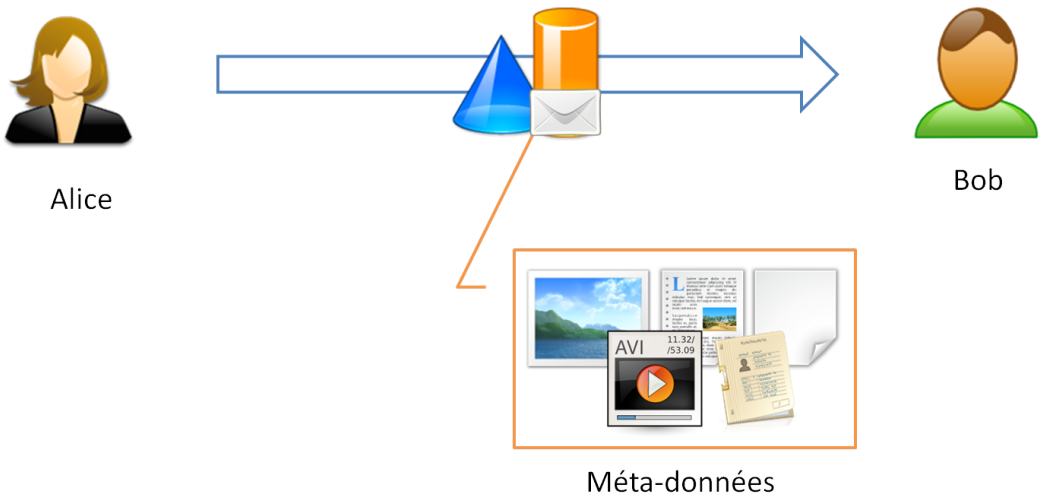


FIGURE 2.9 – Illustration d'un exemple d'application d'enrichissement. Reprenons Alice et Bob. Alice veut fournir du contenu supplémentaire dans le signal qu'elle diffuse. L'intérêt de l'enrichissement des données en utilisant l'insertion de données cachées est de fournir un seul document de taille égale à la donnée originale mais avec des informations supplémentaires. L'objectif est de cacher un maximum d'information sans dégrader le signal original. A la lecture du fichier, Bob pourra lire le signal ainsi que toute les informations complémentaires dissimulées.

Nous voulons toujours garder la marque la plus invisible possible, avec une capacité raisonnable.

2.3.4 Fingerprinting

Dans un état d'esprit similaire à la protection des documents, nous pouvons imaginer des contraintes identiques avec un cas d'utilisation différent. Imaginons la diffusion d'un document multimédia protégé par insertion d'un message avec un cadre juridique reconnaissant ce type de protection. Si au lieu d'insérer des informations de droit d'auteur, nous déposons des informations sur l'acheteur nous arrivons à un cas d'utilisation intéressant (figure 2.11).

La donnée cachée étant protégée de certaines déformations, en cas de distribution non-autorisée nous pouvons tracer l'acheteur de la donnée et donc le responsable de ce partage illégal. Si la fraude est organisée, non pas par un utilisateur mais par un groupe d'utilisateurs malveillants, ils peuvent s'unir et produire un nouveau contenu qui potentiellement ne serait pas protégé. Ces attaques sont des attaques par collusion [Kiyavash et Moulin, 2006].

L'objectif est de mettre en place un processus de tatouage robuste, mais qui doit

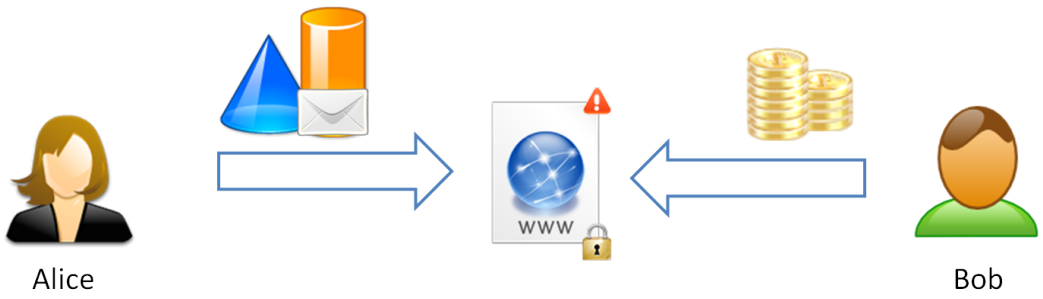


FIGURE 2.10 – Illustration d'un exemple d'application de protection. Supposons qu'Alice veut mettre à disposition un document de manière sécurisée. Moyennant finance, Bob peut obtenir ce document tout en sachant qu'une marque est dissimulée dans le média. Ces informations peuvent être des informations sur les droits d'auteur d'Alice par exemple. En cas d'utilisation illicite du signal d'Alice (revente, modification du contenu, etc.), l'objectif est de prouver que le document est initialement la propriété d'Alice et que sa distribution est réglementée. C'est ici que la marque insérée intervient, elle sert à prouver que ce signal est utilisé de manière illégale. Afin qu'une telle chose soit possible, il faut que la marque reste présente dans le signal pour un ensemble de modifications prises en considération.

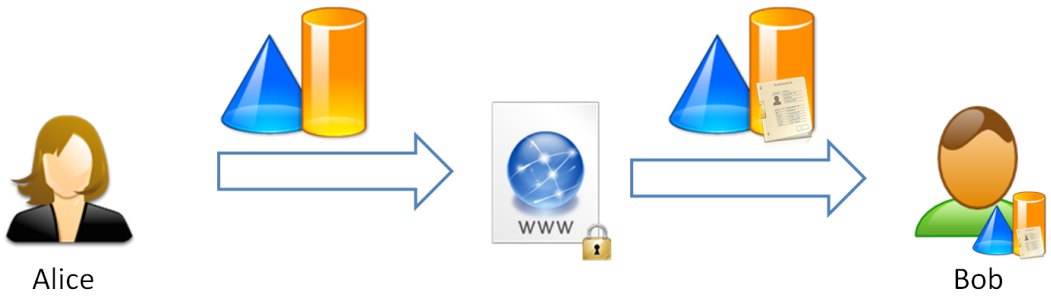


FIGURE 2.11 – Illustration d'un exemple d'application de traçage de traitre.

prioritairement contrecarrer ce genre d'attaque. De récentes méthodes dans la traque de traitre (*fingerprinting*) ont été développées autour de cette thématique en s'appuyant sur les codes de Tardos [Tardos, 2008].

Le problème de ces genres de protection est que, de nos jours, elles ne peuvent pas encore être reconnues d'un point de vue juridique.

2.3.5 Contrôle d'intégrité

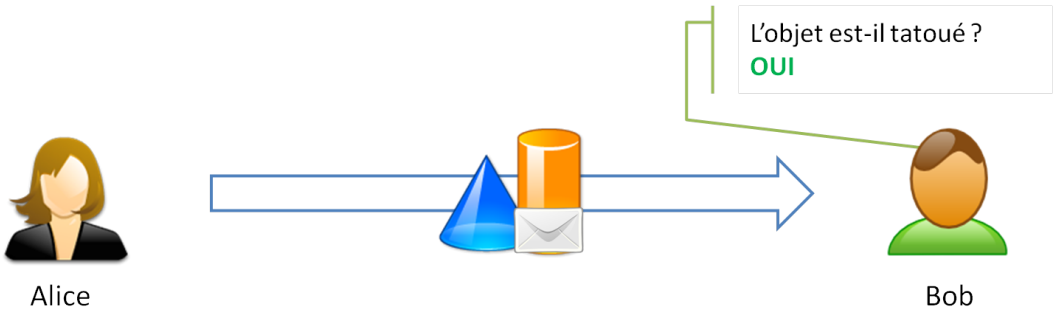


FIGURE 2.12 – Illustration d'un exemple d'application de contrôle d'intégrité. Alice envoie un signal tatoué à Bob. Dans le cas d'un contrôle d'intégrité, si Bob réussit à récupérer le message ou encore détecter sa présence alors la donnée n'a pas été modifiée. Le cas échéant, la donnée a été modifiée.

L'un des rares cas où nous souhaitons l'insertion fragile d'un message est pour des applications de contrôle d'intégrité. Authentifier un document, savoir s'il a été modifié peut être d'une grande utilité sur des données sensibles, ou précises. Des méthodes complémentaires ne s'appuyant pas sur l'insertion de données cachées, comme le calcul d'un hash sont également possibles.

A l'instar des signatures numériques, une nouvelle volonté d'application émerge autour des signatures perceptuelles [Shuo-zhong et Xin-peng, 2007]. Comme nous le savons, pour une signature numérique ou pour le calcul d'un hash, la moindre modification entraine un résultat totalement différent. Si nous allégeons la contrainte des données parfaitement identiques, mais qu'on tolère une erreur dans les données telle que le signal reste similaire pour avoir une signature identique alors nous parlons de signature numérique perceptuelle.

C'est dans ce cadre que peut intervenir l'aspect de l'insertion de données cachées. Ainsi si les données sont similaires la marque insérée restera au sein du signal qui sera considéré comme valide (figure 2.12). Le cas échéant la marque disparaît.

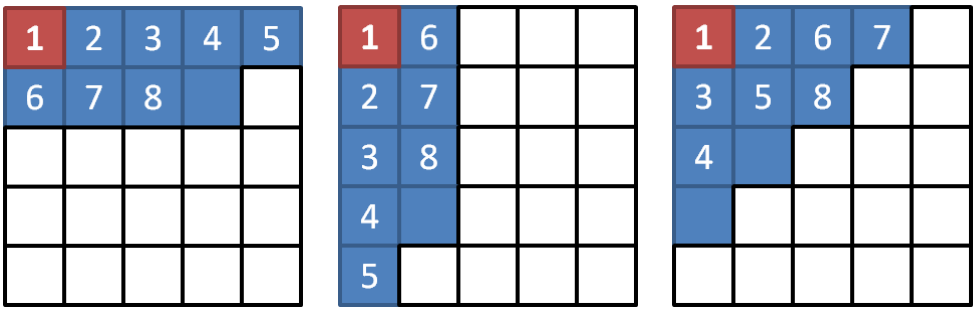
2.4 Synchronisation 3D

La synchronisation est une phase majeure dans le processus de tatouage et d'extraction. Le principe est de repérer dans le signal des zones favorables à l'insertion d'information, savoir les retrouver à l'identique après l'insertion du message et pouvoir les parcourir de manière unique et toujours dans le même ordre (figure 2.5).

Il est important de remarquer pour chacune des méthodes sa phase d'initialisation, la sélection et le parcours des zones choisies, ainsi que son comportement après la phase d'insertion. La phase de synchronisation doit se suffire à elle-même. C'est-à-dire, que même si le maillage n'est pas modifié, nous devons pouvoir retrouver les mêmes zones indépendamment du message inséré.

Sur des signaux audio, ou sur des images il est assez simple de synchroniser le signal hôte. Tout d'abord sur les signaux audio, dans le domaine temporel accompagné d'une horloge nous pouvons parcourir le signal de manière périodique, ou encore de manière pseudo-aléatoire en fonction d'une clé connue par les utilisateurs.

Sur des images 2D, il est facile de parcourir de manière unique l'ensemble de l'image. Soit en lisant l'image ligne par ligne (2.13.a.), ou colonne par colonne (2.13.b.) ou encore en zigzag (2.13.c.) ou de manière pseudo-aléatoire.



a. b. c.

FIGURE 2.13 – Exemples de synchronisations sur des images 2D : a) par un parcours ligne par ligne ; b) colonne par colonne ; c) en zigzag.

Nous allons donc nous intéresser aux méthodes de synchronisation lors de l'insertion de données cachées sur des maillages surfaciques. Comme la plupart des méthodes sont réalisées dans le domaine spatial, nous proposons de classifier les différentes méthodes de la façon suivante :

— Synchronisation non informée ;
— Synchronisation géométrique ;
— Synchronisation topologique ;
— Synchronisation sur des zones caractéristiques ;

— Synchronisation sur des structures de données ;
— Synchronisation dans un domaine fréquentiel ;
— Synchronisation dans un domaine ondelette ;

2.4.1 Synchronisation non-informée

Le tatouage non informé ne prend pas en compte les informations liées ni à la géométrie, ni à la topologie du maillage. L'insertion du message se fait par des jeux de permutation dans la liste des sommets, dans la liste des facettes ou encore dans la liste des sommets qui composent la facette.

2.4.2 Synchronisation particulière

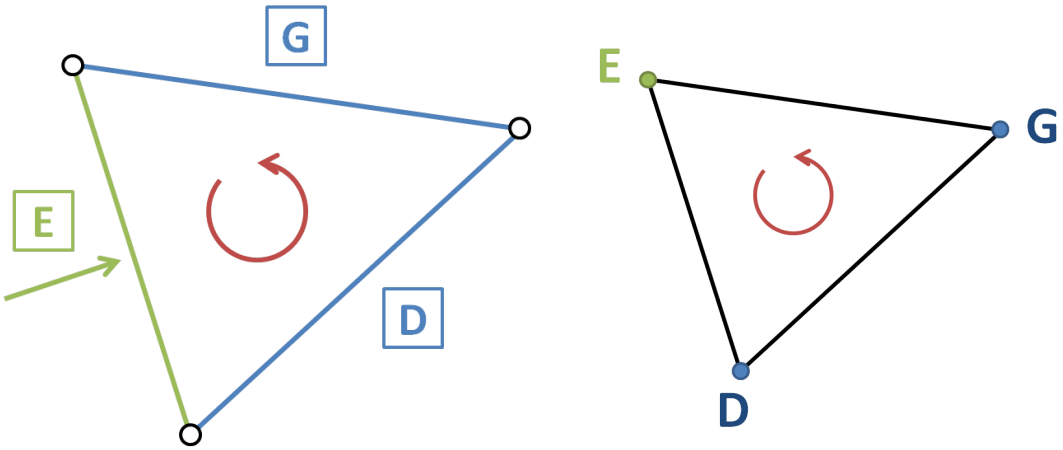
L'une des premières méthodes de dissimulation d'information proposée [Ohbuchi *et al.*, 1997] se sert du message pour synchroniser le maillage. Le principe est de créer une suite de triangles qui sera fonction du message. Initialement, les auteurs orientent tous les triangles de la même façon. Puis pour commencer le parcours, ils doivent choisir un triangle et une arête.

Grâce à l'orientation, nous définissons une arête "gauche" et une arête "droite" comme l'illustre la figure 2.14.a. L'idée est donc de parcourir le maillage en fonction du message. Ainsi si nous voulons coder un 0, nous sélectionnerons le triangle voisin qui partage l'arête "droite" par exemple. Et réciproquement pour coder un 1, nous choisissons le triangle qui partage l'arête "gauche". Et ainsi jusqu'à ce que le message soit complètement lu.

Lors de l'insertion, la bande de triangles parcourue est déconnectée du maillage (figure 2.15). C'est-à-dire, tous les triangles qui composent la bande sont supprimés, et les sommets qui la composent sont dupliqués pour conserver la frontière de la bande de triangles. Ainsi il y a un trou dans le maillage. A ce maillage troué, on rajoute la bande supprimée en ne la connectant pas à l'arête initialement choisie lors du parcours.

Cette technique a la particularité de se servir du message lors de la synchronisation. Si on fait abstraction des problèmes sur la topologie du maillage, en effet avec cette méthode il est impossible de coder un message ayant une succession importante de 0 ou de 1 ; lors de l'extraction il faut pouvoir retrouver correctement le message.

Le problème principal est que la nature topologique du maillage est différente et n'est pas forcément compatible avec les formats de fichiers. Cette cassure dans la topologie du maillage peut ne pas être reconnue suivant le format, il serait donc impossible de retrouver correctement la bande construite. Le cas échéant, il suffit de repérer dans le maillage la connexion entre la bande de triangle et le maillage, puis de lire le parcours effectué pour récupérer le message.



a. b.

FIGURE 2.14 – a) Parcours des arêtes du maillage. Avec une orientation donnée, et une arête initialement choisie, le maillage peut être parcouru de la façon suivante. Soit E l'arête courante et posons D l'arête qu'on rencontre immédiatement lorsqu'on parcourt le triangle dans le sens de l'orientation et G la deuxième. Par convention D sera l'arête de droite et G celle de gauche. Une fois l'arête sélectionnée elle devient l'arête entrante du triangle voisin. Le processus continue jusqu'à la fin du parcours, suivant les règles données par la méthode. b) Parcours des sommets du maillage. De la même façon qu'avec a), mais avec les sommets. Dans ce cas il est important de préciser le triangle dans lequel on se trouve, et les règles de passage d'un triangle à l'autre.

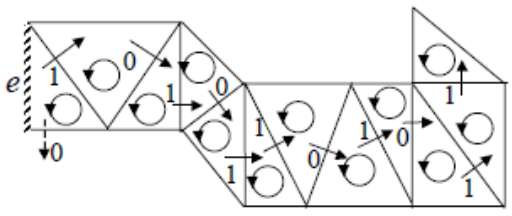


FIGURE 2.15 – Représentation d'une bande de triangles codante.

2.4.3 Synchronisation géométrique

La synchronisation se base uniquement sur la géométrie par des méthodes de regroupement des points en *cluster* [Agarwal et Prabhakaran, 2007]. Nous avons vu qu'il n'est pas toujours évident d'établir un ordre de lecture des sommets. Tout d'abord les sommets sont partitionnés par des heuristiques de type *"divide and conquer"* [Brassard et Bratley, 1996], nous obtenons ainsi des groupes de points construits en fonction de leur voisinage. Ainsi, un graphe reliant les points aux sommets les plus proches est construit. En supposant une distance maximale fixée, si la distance entre deux points est inférieure alors une arête est ajoutée dans le graphe. A partir de ce graphe, les composantes connexes correspondent aux clusters de la méthode, nous représentons un exemple de ces clusters sur la figure 2.16.

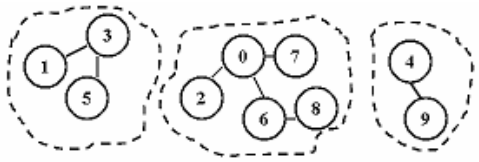


FIGURE 2.16 – Représentation d'un ensemble de clusters de sommets.

Dans un cluster, les auteurs distinguent deux classes de points : des points d'ancrage qui serviront de repères et des points codant qui seront déplacés pour dissimuler le message. Les points d'ancrage sont ceux qui ont un degré supérieur à 2 dans le graphe construit par la méthode heuristique. Les sommets codant sont donc ceux qui ont un degré égal à 1 (cf. figure 2.17).

Le problème est maintenant de construire l'ordre des points dans un cluster et dans un deuxième temps l'ordre des clusters. L'ordre des points dans le cluster se fait par un tri en fonction du degré du point d'ancrage du cluster et du nombre de bits insérés par le déplacement du point codant. L'ordre des clusters est établi en fonction du nombre de sommets qui le compose et la distance entre les points d'ancrage au sein d'un même cluster.

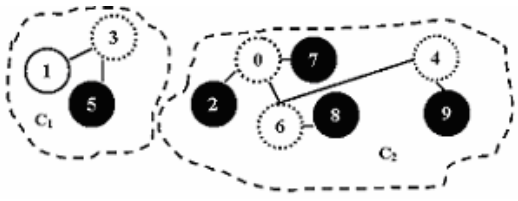


FIGURE 2.17 – A partir des clusters, les sommets codants sont représentés en noir et les sommets d'ancrage sont en pointillés.

L'avantage de la méthode est qu'elle ne repose pas sur la triangulation, ainsi toutes les attaques sur le remaillage est inefficace. De plus, la méthode repose sur les distances Euclidiennes, ainsi la méthode est robuste aux transformations géométriques conservant les rapports de distances.

2.4.4 Synchronisation topologique

L'objectif des méthodes qui suivent est de parcourir le maillage par un ruban de triangles comme si nous parcourions une image par ses lignes ou ses colonnes. Remarquons immédiatement que ce genre de synchronisation est fortement dépendant du maillage, un simple remaillage désynchronise et entraîne la perte de la marque insérée. La deuxième faille est l'initialisation. Il faut caractériser de manière unique le premier triangle qui servira de départ à la lecture du ruban. Choisir le triangle qui a la plus grande / petite aire, le plus grand / petit périmètre, sont des critères possibles [Benedens, 1999] mais qui peuvent varier en fonction des déformations que l'objet subit.

Connaissant ces limites aux méthodes de parcours de ruban de triangle, la stéganographie (ou communication secrète) est une application particulièrement adaptée à ce genre de techniques. Les auteurs doivent donc supposer que l'objet ne subira pas de déformations non désirées.

A présent, nous avons un point de départ de lecture du ruban. Il faut maintenant se donner des règles pour le parcourir. A partir de la méthode décrite précédemment, en sélectionnant une arête d'entrée dans le triangle, il reste deux possibilités pour sortir du triangle. Au lieu de définir l'ordre en fonction du message, plusieurs solutions sont possibles :

— Le parcours prédéfini est déterministe [Mao *et al.*, 2001] ;

— Une séquence pseudo-aléatoire définit l'ordre des triangles à parcourir [Macq et Cayre, 2003] [Tournier *et al.*, 2011a] ;

Si nous regardons dans d'autres disciplines, nous trouvons des méthodes de parcours intéressantes. Notamment en compression, avec l'algorithme Edgebreaker [Rossignac, 1999]. Dans cette technique, nous nous intéressons au parcours du maillage (figure 2.18). A partir d'un ensemble de configurations préalablement établie, l'auteur parcourt le maillage en fonction du cas qu'il rencontre. Lors de cette première étape, il enregistre l'historique de son parcours dans le maillage par la liste des sommets accompagnés de leurs coordonnées, la configuration rencontrée lors du passage sur ce triangle. Une fois l'historique construit, il peut obtenir à la fois un ordre sur les triangles mais aussi sur les sommets, comme le montre ce simple exemple sur la figure 2.18.

Cet algorithme est utilisé en stéganographie [Bogomjakov *et al.*, 2008], lors de la phase de synchronisation afin de déterminer un ordre sur les triangles.

Cet ordre est unique et ne dépend que du triangle choisi initialement, ce qui est une propriété essentielle pour une technique de synchronisation. Ainsi, une telle méthode

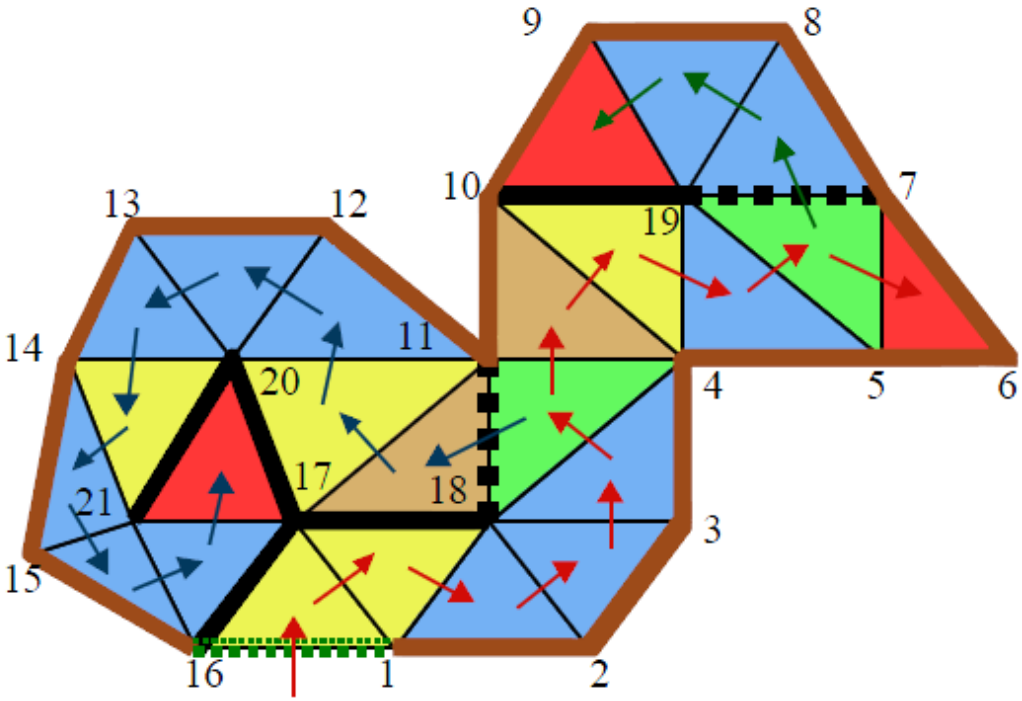


FIGURE 2.18 – Illustration du parcours des sommets et des arêtes d'un maillage triangulaire par l'algorithme Edgebreaker.

peut être utilisée avec des techniques de tatouage qui ne modifient pas la topologie de l'objet.

Dans un même ordre d'idée, mais cette fois dans le domaine de la CAO [Béniere *et al.*, 2010], le principe est de construire des grilles les plus régulières possibles dans le maillage. Cette décomposition est appelée décomposition en carreaux quadrangulés, où un quadrangle est une paire de triangles adjacents. Les auteurs procèdent en trois étapes :

— Pour chaque quadrangle, un coefficient de qualité est estimé. Il est fonction des angles aux sommets et de l'angle dièdre. Plus le quadrangle est proche d'un rectangle, meilleur est son coefficient de qualité.

— A partir du quadrangle de meilleur qualité, les zones quadrangulées sont construites par propagation. A chaque étape, le quadrangle voisin ayant le meilleur score est ajouté à la zone quadrangulée. Le processus s'arrête lorsqu'il ne reste plus de triangle dans le voisinage ou si les quadrangles restants sont de mauvaise qualité.

— La zone quadrangulée est ensuite découpée en polygones rectilignes de façon à

indexer chaque quadrangle par une position unique sous la forme d'un numéro de ligne et d'un numéro de colonne comme nous le ferions pour une image 2D.

Cette décomposition en polygones rectilignes peut être intéressante. Seulement pour compléter la méthode, dans un maillage composé de plusieurs polygones rectilignes, il faut être capable de les ordonner. En supposant cette étape résolue, il faut également lors de l'insertion préserver la configuration de la qualité des quadrangles pour retrouver exactement le même résultat lors de la synchronisation.

2.4.5 Synchronisation sur des zones caractéristiques

Lors de l'insertion d'un message, quelle que soit la méthode d'insertion, il y a des zones dans le maillage qui sont plus propices à une déformation que d'autres. Pour des raisons de robustesse ou d'imperceptibilité, il est parfois préférable de sélectionner de manière plus appropriée des zones caractéristiques dans le maillage. Ces zones peuvent être un ensemble de points, de facettes ou de triangles.

Une des premières méthodes de sélection de points se basait sur la distance des points par rapport à ses voisins directs dans le maillage. Si la distance entre le point et son voisinage est inférieure à un certain seuil alors il est sélectionné [Harte et Bors, 2002]. Une fois la sélection faite, les sommets sont triés de manière croissante en fonction de cette distance calculée.

Nous remarquons que ce genre de technique est très fragile à toute perturbation géométrique, mais elle fait partie des premières méthodes de sélection de zones caractéristiques.

Pour résister à des attaques de découpe, ou de remplacement de parties de maillage, l'idée est de créer des patchs dans lesquels nous insérons dans chacun la même information. Pour créer ces patchs, nous nous tournons vers les domaines de recherche en segmentation. Dans [Sales *et al.*, 2011] et [Cayre *et al.*, 2003], les auteurs insistent particulièrement sur la méthode de synchronisation et proposent une méthode de création de patchs à partir de points caractéristiques.

Pour extraire les points caractéristiques, ils se basent sur la détection de maxima locaux. Ils sont indépendants de la pose et restent invariants par des opérations de mise à l'échelle. A partir de ces points, les auteurs [Sales *et al.*, 2011] construisent les patchs qui serviront à l'insertion. Pour éviter les chevauchements, ils déterminent les cellules de Voronoi géodésique à partir de l'ensemble des points caractéristiques. Puis pour chaque point, ils prennent le plus grand n voisinage tel qu'il reste strictement inclus dans la cellule de Voronoi associée ou en limitant le nombre de points contenu dans le patch.

Une fois les patchs construits, ils ordonnent les sommets qui le composent du plus proche au plus éloigné en fonction de sa distance géodésique avec le point caractéristique.

2.4.6 Synchronisation sur des structures de graphe

Comme nous le voyons dans cette section, il est difficile de trouver un moyen de parcourir un ensemble de points et de triangles dans un espace métrique 3D. Une méthode originale a été proposée [Amat *et al.*, 2010] ne déplaçant aucun sommet et utilisant une structure issue de la théorie des graphes, l'arbre couvrant de poids minimum euclidien (ACPME). Avec [Mao *et al.*, 2001; Ohbuchi *et al.*, 1997] c'est l'une des seules méthodes de dissimulation d'information qui ne modifie pas la position des points, et donc la seule du domaine du tatouage numérique puisque les méthodes citées soit augmentent la taille de la donnée initiale en rajoutant des points [Mao *et al.*, 2001], soit n'ont pas une méthode de synchronisation indépendante du message [Ohbuchi *et al.*, 1997].

Nous nous intéressons à cette méthode de synchronisation [Amat *et al.*, 2010] car elle possède de très bonnes propriétés. D'une part, si nous disposons d'une orientation, d'une racine et que la construction de l'arbre couvrant repose sur la distance entre les points du maillage, alors la structure de l'ACPME est unique. En partant de la racine, le parcours de l'arbre est possible. Nous obtenons donc une relation d'ordre sur l'ensemble des sommets de l'arbre. Ainsi, en ne déplaçant aucun point lors de l'insertion, les connexions dans l'ACPME sont conservées. Pour toutes les raisons évoquées précédemment, cette méthode de synchronisation repose sur des outils qui ont de bonnes propriétés.

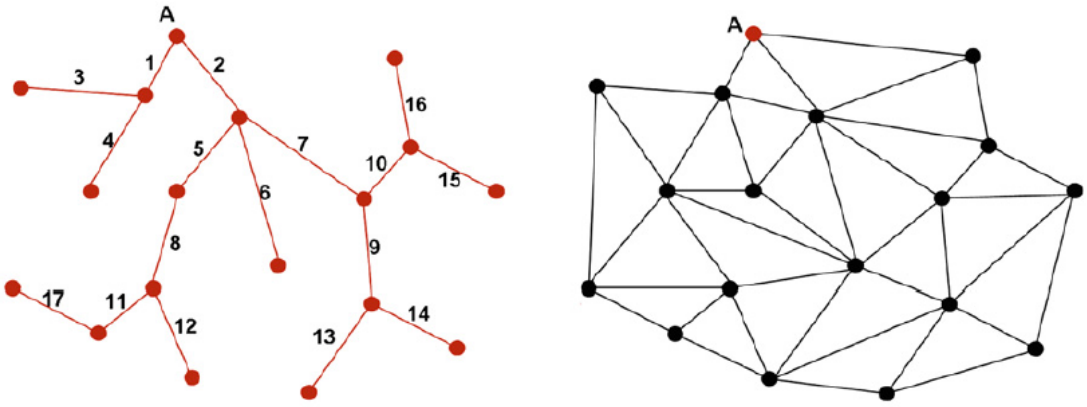


FIGURE 2.19 – Sur la gauche, une représentation de la construction d'un ACPME en prenant comme base les points correspondant au maillage représenté sur la droite, et pour origine le point A.

A partir de cet outil, pour préparer l'insertion du message, ils recherchent dans l'arbre les sommets de degré 4. A partir de ce sommet ils sélectionnent le quadrangle, deux triangles partageant une même arête. L'insertion repose sur les flips d'arêtes dans les quadrangles si le quadrangle n'est pas modifié alors un 0 est codé, s'il y a eu un flip d'arête

dans le quadrangle alors nous avons codé un 1. Dans le cadre de la synchronisation, les quadrangles doivent vérifier les conditions suivantes :

— Coplanarité : nous chercherons à sélectionner les quadrangles aussi plans que possible, ainsi après insertion, le changement de diagonale du quadrangle restera plan et n'entrainera pas de déformation visuelle visible ;

— Convexité : pour éviter de créer des anomalie sur la topologie du maillage et conserver la propriété manifold du maillage, les quadrangles doivent être convexes ;

— Recouvrement : pour éviter les erreurs de synchronisation, nous éviterons de sélectionner deux quadrangles voisins.

L'un des principaux inconvénients de la méthode est sa fragilité et sa complexité algorithmique pour les données qui sont traitées et les conditions d'utilisation. D'un point de vue robustesse, la fragilité de la méthode limite la condition d'utilisation au contrôle d'intégrité par exemple. Du fait des bonnes propriétés des outils utilisés lors de la synchronisation, cette méthode est le point de départ de l'ensemble des travaux qui sont présentés dans cette thèse.

Pour l'ensemble des méthodes présentées précédemment, la synchronisation et l'insertion se font dans le domaine spatial. Tout comme en imagerie, ou en audio, il est possible de transformer le signal dans un autre domaine ou de le représenter d'une façon différente. En imagerie, nous avons deux autres domaines : le domaine fréquentiel avec les transformées de Fourier par exemple et le domaine des ondelettes (Haar, Daubechies, etc.).

2.4.7 Synchronisation dans un domaine fréquentiel

L'idée de passer dans un domaine fréquentiel est, comme en image, pour pouvoir gagner en terme d'invisibilité mais aussi de robustesse. La première décomposition spectrale [Karni et Gotsman, 2000] a été présentée en vue de transmettre progressivement la géométrie du maillage. Ce genre de technique commence à se mettre en place dans le domaine du tatouage 3D qui reste encore une science jeune.

Nous trouvons dans la littérature une méthode pour la 3D analogue à la DCT pour les images 2D [Cayre *et al.*, 2003] utilisant la décomposition spectrale de la géométrie ou plus simplement [Ohbuchi *et al.*, 2001] en passant par la matrice de Kirchhoff K définie par :

$$K = D - A,$$

avec D défini par : $\forall i \in \mathbb{N}\ i < N\ d_{i,i}$ est égal au nombre de voisins du i^e sommet et $\forall (i,j) \in \mathbb{N}^2\ i,j < N\ i \neq j\ d_{i,j} = 0$; et A la matrice d'adjacence $\forall (i,j) \in \mathbb{N}^2\ i,j < N$ $a_{i,j} = 1$ si les sommets v_i et v_j sont voisins, sinon $a_{i,j} = 0$. Nous notons λ_i, les valeurs propres de la matrice K, et e_i les vecteurs propres normés associés.

Dans ces méthodes, la synchronisation est le passage du système de coordonnées cartésiennes à un système de cordonnées spectrales. L'insertion du message se fait par étalement de spectre, une technique connue dans le domaine pour résister à l'ajout de bruit.

2.4.8 Synchronisation dans un domaine ondelette

Ce genre de représentation est à la fois spatiale et fréquentielle.

A partir d'un signal original, nous construisons un signal de résolution inférieure obtenu par sous échantillonnage, lissage, filtrage, par exemple. Pour éviter une perte d'information, celle-ci se retrouve dans les vecteurs ondelettes (figure 2.20). En d'autres mots, à partir du signal de qualité inférieure, lorsqu'on rajoute l'information contenue dans les vecteurs ondelettes nous retrouvons un signal d'aussi bonne qualité que l'original. Remarquons que le signal n'est pas parfaitement égal à l'original, cette opération de décomposition n'est généralement pas réversible.

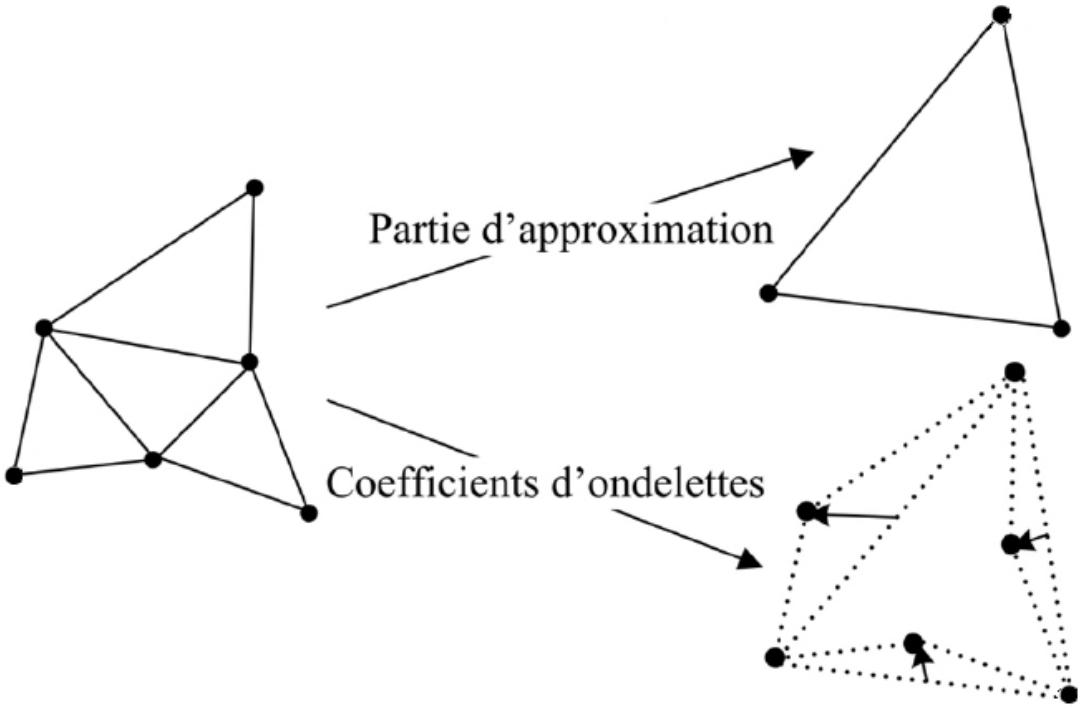


FIGURE 2.20 – Ondelettes 4-1 sur les maillages triangulés.

Dans [Wang *et al.*, 2008], la décomposition ondelette est utilisée pour travailler sur le maillage de basse résolution. Pour leur méthode de synchronisation la plus simple, ils se basent sur la décomposition en ondelettes et le tri des arêtes du maillage d'approximation

par leur longueur. L'insertion se fait par quantification de la longueur des arêtes qui seront tatouées, par une méthode du type "Scalar Costa" [Eggers *et al.*, 2003].

Nous pouvons également nous servir des vecteurs ondelettes pour dissimuler de l'information. Dans [Kim *et al.*, 2005] après décomposition dans le domaine des ondelettes, le vecteur $d_i = (x_i, y_i, z_i)$ subit une deuxième transformation par un changement de système de coordonnées. D'une représentation dans le système de coordonnées cartésiennes, le vecteur est exprimé dans le système sphérique $d_i = (|d_i|, \theta_i, \phi_i)$. L'insertion se fait par étalement de spectre sur la norme de d_i dans la direction d'une porteuse générée pseudo-aléatoirement.

Malheureusement dans ces méthodes dans le domaine des ondelettes, la synchronisation est simplement un jeu de réécriture du signal. Combiner ce type de méthode à la recherche de zones caractéristiques pourrait être intéressant.

2.5 Conclusion

Dans le cadre de cette thèse nous nous intéresserons essentiellement à cette phase de synchronisation. Il est clair que ceci peut être frustrant de ne pas aller au bout du processus complet d'insertion de la donnée, mais il est également important de se focaliser sur ce genre de problématique qui est essentiel dans toute méthode.

Nous avons essayé de proposer une nouvelle façon de voir le tatouage par cette organisation, ne mettant plus en avant l'aspect de robustesse de la méthode mais en les classifiant uniquement par leur synchronisation. Il se peut qu'en changeant ce point de vue, nous révélons des méthodes d'insertion qui sont robustes mais à condition d'avoir en amont un système de synchronisation respectant les mêmes conditions de robustesse, prouvant que la synchronisation doit être fortement corrélée avec le processus d'insertion. Parmi ces familles de méthodes de synchronisation, nous avons fait ressortir deux familles celles qui se font dans le domaine spatial, et celles qui font appel à un changement de domaine.

La richesse des méthodes de synchronisation se trouve essentiellement dans le domaine spatial, ce qui est logique si nous suivons l'évolution du tatouage en image et en vidéo. En effet, l'utilisation des domaines transformés n'est venu que plus tard. Nous avons les méthodes qui se basent sur la topologie de l'objet, c'est-à-dire les connexions entre les sommets ; la recherche de zones caractéristiques, également appelées amers ou points d'ancrage ; et de manière assez originale l'utilisation de structures de graphe.

Chapitre 3

Théorie des graphes

Préambule

L'histoire de la théorie des graphes débute avec les travaux d'Euler au XVIII^{e} *siècle et trouve son origine dans le célèbre problème des ponts de Koenigsberg [Euler, 1736] - problème de coloration de graphes - où Euler se demandant s'il était possible de parcourir tous les ponts de la ville une seule et unique fois sans passer deux fois par le même pont.*

Également connu pour le théorème des quatre couleurs [Cayley, 1879], un problème pratique des cartographes cherchant le minimum de couleurs nécessaires pour colorer une carte sans que deux éléments juxtaposés soient de la même couleur.

Aujourd'hui, les graphes modélisent toutes sortes de problèmes dans diverses disciplines telles que la chimie, la biologie, les sciences sociales, l'économie, l'informatique, etc.

Sommaire

3.1 Introduction

Dans le chapitre précédent, nous avons présenté un état de l'art sur la synchronisation dans les maillages. Dans le cadre de cette thèse, nous nous intéressons particulièrement aux méthodes de synchronisation par structure de données, notamment les

arbres couvrant de poids minimum Euclidien (ACPME). Dans ce chapitre nous allons placer les problèmes du calcul des arbres couvrant de poids minimum (ACPM) et les notions nécessaires pour l'utilisation de cet outil dans nos problèmes de dissimulation d'information.

3.2 Graphes

Les graphes sont des outils généralement utilisés pour modéliser un problème. Ils sont souvent utilisés en réseau, télécommunication, informatique, mais également en biologie, science sociale, économie, et bien d'autres domaines.

Avant de présenter les ACPM, nous consacrons cette section à la définition des graphes (3.2.1) et des notions de base nécessaires. En section 3.2.2 nous développons les propriétés de connexité. Pour finir, (3.2.3), nous introduirons les structures basées sur la notion de voisinage ce qui nous amènera à la définition des ACPM.

3.2.1 Définitions et analogie avec la 3D

Un graphe est une structure mathématique assez simple. C'est un ensemble de sommets connectés. Nous notons $G = (V_G, E_G)$ le graphe avec V_G l'ensemble des sommets du graphe et E_G l'ensemble des connexions. A partir de ces données, nous pouvons distinguer différents types de graphes en fonction de deux propriétés : l'orientation et la valuation.

Nous appelons connexion, une liaison entre deux sommets. Si la connexion est orientée, il s'agit d'un arc ; sinon c'est une arête. Un arc possède une origine $v_i \in V_G$ et une extrémité $v_j \in V_G$, nous le notons (v_i, v_j). Ce qu'il faut retenir c'est que les arcs (v_i, v_j) et (v_j, v_i) sont différents, ils ont une orientation opposée l'un à l'autre. Les arêtes e_i sont des paires de sommets, ainsi les arêtes $\{v_i, v_j\}$ et $\{v_j, v_i\}$ sont les mêmes.

Chaque connexion peut être pondérée par une valeur appelée poids que nous notons $\omega(e_i) \in \mathbb{R}$. Nous parlons de graphe valué si tel est le cas et notons $\omega : E_G \to \mathbb{R}$ la fonction de valuation.

Dans notre cas, nous nous intéressons exclusivement aux graphes non-orientés valués et telles que les valeurs des arêtes soient strictement positives. Par convention nous notons $V_G = \{v_1, v_2, ..., v_n\}$ l'ensemble des n sommets et $E_G = \{e_1, e_2, ..., e_m\} \subset V_G^2$ les m arêtes du graphe G.

Exemple. Sur la figure 3.1 qui illustre un graphe à 9 sommets et 16 arêtes. On a : $V_G = \{A, B, ..., I\}$ et $E_G = \{\{A, B\}, \{A, D\}, ..., \{H, I\}\}$.

A partir de la définition du graphe on peut immédiatement construire une analogie entre un maillage 3D triangulé et un graphe. Notons $\mathcal{M}$ un maillage triangulé. Comme nous l'avons vu dans le chapitre précédent, $\mathcal{M}$ est composé de différentes primitives :

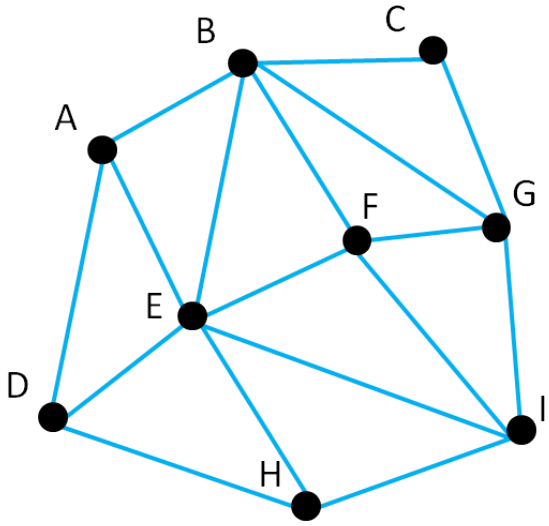


FIGURE 3.1 – Un exemple de graphe non-valué non-orienté.

des points $v_i \in \mathcal{V}_\mathcal{M}$, $\mathcal{V}_\mathcal{M} \subset \mathbb{R}^3$, des connexions $e_i \in \mathcal{E}_\mathcal{M}$, $\mathcal{E}_\mathcal{M} \subset {\mathcal{V}_\mathcal{M}}^2$ et des triangles $t_i \in \mathcal{T}_\mathcal{M}$, $\mathcal{T}_\mathcal{M} \subset {\mathcal{E}_\mathcal{M}}^3$.

Ainsi, on fait correspondre un point du maillage $v_i \in \mathcal{V}_\mathcal{M}$ à un sommet $v_i \in V_G$ et respectivement une connexion $e_i \in \mathcal{E}_\mathcal{M}$ à une arête $e_i \in E_G$ dans le graphe $G = (V_G, E_G)$. Concernant la fonction de poids ω_G, on associe la distance euclidienne d_2 définie sur $\mathbb{R}^3$ tel que $\omega_G(e_i) = d_2(e_i)$.

	Graphe $G = (V_G, E_G, \omega_G)$	Maillage $\mathcal{M} = (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M}, \mathcal{F}_\mathcal{M})$
$d = 0$	$v_i \in V_G$	$v_i \in \mathcal{V}_\mathcal{M}$
$d = 1$	$e_i \in E_G$, $e_i = \{v_i, v_j\}$	$e_i = \{v_i, v_j\} \in \mathcal{E}_\mathcal{M}$
$d = 2$		$t_i = \{e_i, e_j, e_k\} \in \mathcal{F}_\mathcal{M}$
Poids	$\omega_G(e_i) \in \mathbb{R}^+$	$d_2(e_i) \in \mathbb{R}^+$

TABLE 3.1 – Tableau récapitulatif de l'analogie entre un maillage 3D et un graphe.

3.2.2 Connexité et notion de voisinage

A partir des connexions d'un graphe G, nous pouvons récupérer des informations locales sur le voisinage d'un sommet. Pour un sommet $v_i \in V_G$, par convention nous notons $N(v_i)$ son voisinage, c'est-à-dire l'ensemble des sommets qui sont reliés à v_i. Dans un maillage le nombre de voisins connectés à un point est appelé valence ; dans ce cadre, il est appelé degré et nous avons $d(v_i) = |N(v_i)|$.

De manière plus globale, nous nous intéressons aux propriétés de connexité d'un graphe. Pour cela nous avons besoin des notions de chemin, de cycle et de composante connexe.

Simplement, un chemin est un ensemble d'arêtes qui relient un sommet à un autre. Nous parlons de chemin de longueur $k \in \mathbb{N}$, lorsque le chemin reliant un sommet v_i à v_j a besoin d'au moins k arêtes. Dans cette notion, la valuation des arêtes ne joue aucun rôle. Formellement, nous pouvons écrire que le chemin $P(v_i, v_j)$ de longueur $k \in \mathbb{N}$ reliant un sommet $v_i \in V_G$ à $v_j \in V_G$, $v_j \neq v_i$ est un ensemble d'arêtes tel que :

$$\exists (z_1, z_2, ..., z_k) \in V_G^k, \forall i < k \{z_i, z_{i+1}\} \in E_G : P(v_i, v_j) = \{\{v_i, z_1\}, \{z_1, z_2\}, ..., \{z_k, v_j\}\}.$$

Exemple. Dans l'exemple, illustré figure 3.1, il existe un chemin de longueur 2 entre A et C : $P(A, C) = \{\{A, B\}, \{B, C\}\}$.

Un cycle est un chemin particulier de longueur $k > 1$ reliant un sommet $v_i \in V_G$ à lui-même.

Exemple. Sur la figure 3.1 :

- $P(A, A) = \{\{A, B\}, \{B, E\}, \{E, A\}\}$ est un cycle de longueur 3 ;
- $P(A, A) = \{\{A, E\}, \{E, I\}, \{I, H\}, \{H, D\}, \{D, A\}\}$ est de longueur 5.

Enfin une composante connexe est une classe d'équivalence pour la relation de connexité. C'est-à-dire, que deux sommets appartiennent à la même classe s'il existe un chemin qui les relie. Un graphe est connexe s'il ne possède qu'une composante connexe. Par la suite nous supposons que le graphe est connexe.

3.2.3 Structures de graphes géométriques

Précédemment nous avons vu, les principales définitions de graphes, de connexions entre les sommets et par conséquent de connexité. Nous avons fait une analogie simple entre les graphes et un maillage. Maintenant nous allons travailler uniquement sur des graphes géométriques. En se basant sur des données géométriques, il n'est pas nécessaire de connaître la définition formelle du graphe géométrique. Les conditions font que les contraintes nécessaires sont largement vérifiées.

A titre indicatif, un graphe est dit géométrique, si la fonction de valuation vérifie l'inégalité triangulaire. C'est-à-dire, s'il existe une arête e_k entre deux points alors le poids de tous les chemins de longueur supérieure ou égale à 2 reliant ces deux points sera supérieur au poids de l'arête $\omega(e_k)$.

Considérons maintenant E_G un ensemble de sommets, issu de données géométriques. Nous allons construire différents types de graphes se basant sur cet ensemble de sommets.

Les graphes de Gabriel (GG)

Les graphes de Gabriel [Gabriel et Sokal, 1969] qu'on note GG sont définis géométriquement de la manière suivante. (u, v) est dans le graphe de Gabriel si et seulement si il n'existe aucun sommet appartenant à la boule de diamètre $\omega(u, v)$. Cette relation est illustrée sur la figure 3.2.

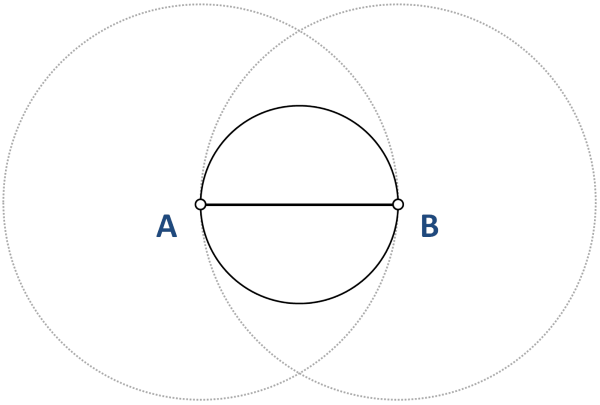


FIGURE 3.2 – L'arête AB est dans le graphe de Gabriel parce qu'il n'y a pas de sommet dans le cercle représenté ci-dessus.

Relative Neighbourhood Graph (RNG)

Le Relative Neighbourhood Graph [Agarwal et Mataušek, 1992] est un graphe qui se définit de la façon suivante :

$$RNG = \{(u, v) \in E : \nexists w \in (N(u) \cap N(v))\}$$

Dans le cas des graphes géométriques, en prenant la distance euclidienne, ce qui est généralement notre cas, l'arête (u, v) du RNG existe si et seulement si il n'y a pas de sommet dans l'intersection des boules de centre u et de rayon $\omega(u, v)$ et celle de centre v et de même rayon. Cette relation est illustrée sur la figure 3.3.

En revenant à notre maillage triangulé, par analogie, on remarque que ce graphe s'obtient en supprimant pour chaque triangle l'arête la plus grande.

De plus, les contraintes étant plus fortes dans les graphes de Gabriel on obtient la relation d'inclusion suivante :

$$RNG \subset GG$$

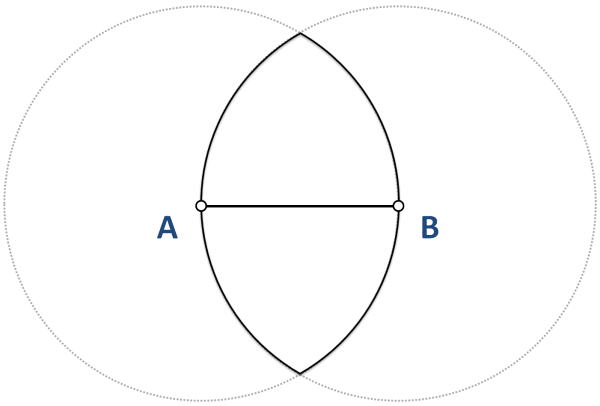


FIGURE 3.3 – L'arête AB est dans le RNG parce qu'il n'y a pas de sommet dans l'intersection des cercles représentés ci-dessus.

Local Minimum Spanning Tree (LMST)

Pour définir ce qu'est un LMST, nous avons besoin de savoir ce que sont les arbres couvrant de poids minimum (ACPM). Pour faire simple, un LMST est la réunion d'arbres couvrants de poids minimum (ACPM) calculéS localement. Pour chaque sommet $v_i \in V$ qui compose le graphe, on sélectionne $N(v_i) \subset V$ les voisins de v_i dans le graphe G. Le LMST est l'union des ACPM calculés sur $N(v_i)$ pour tous les $v_i \in V$. Ce qui donne formellement :

$$LMST(G) = \bigcup_{v_i \in V} ACPM(N(v_i) \cup v_i).$$

il est moins intuitif de déduire une relation entre les graphes précédents, mais nous avons la relation d'inclusion suivante, pour tout graphe :

$$LMST \subset RNG$$

Arbre couvrant de poids minimum

T est un arbre couvrant de poids minimum [Eisner, 1997] (ACPM) si T est un arbre, T connecte tous les sommets et le sommet des arêtes de T est minimum pour la fonction de poids ω_G.

En reprenant en détail, T est un arbre. C'est un graphe connexe sans cycle, on peut le définir de plusieurs manières. Ainsi, les propriétés suivantes sont équivalentes :

— T est un arbre ;

— T est un graphe connexe et $m = n - 1$;

— T est un graphe connexe et la suppression d'une arête le déconnecte ;

— T est un graphe sans cycle avec $m = n - 1$;

— T est un graphe sans cycle et l'ajout d'une arête crée un cycle ;

— $\forall (v_x, v_y) \in V_G^2$, il existe un unique chemin reliant v_x à v_y.

T est un arbre couvrant, c'est-à-dire qu'il connecte tous les sommets ; ou encore qu'il n'existe qu'une seule composante connexe. Aucun sommet n'est isolé.

Enfin, la fonction de poids ω_T écrite de manière formalisée :

$$\omega_T = \sum_{e_i \in E_T} \omega_G(e_i) = \min_{E \subset V^2} \sum_{e_i \in E_G} \omega_G(e_i).$$

Réaliser cette condition, c'est résoudre un problème d'optimisation. A priori, cette réalisation est complexe ; pourtant celle-ci se résout de manière polynomiale. Nous le verrons dans la section suivante avec les algorithmes de Prim [Prim, 1957] et Kruskal [Kruskal Jr, 1956].

Nous notons $T = ACPM(G)$, l'ACPM de G. Dans le cas de la synchronisation sur des données géométrique, l'ACPM est unique en se donnant des règles de construction dans le cas où deux arêtes ont un poids identique. Nous détaillerons plus cette affirmation dans la suite du manuscrit.

3.3 Arbre couvrant de poids minimum

3.3.1 Catégorisation du problème

Le problème de calcul d'arbre couvrant de poids minimum est polynomial [Rajasekaran, 2005]. Trois algorithmes différents le démontrent par construction, il s'agit des algorithmes de Prim [Prim, 1957], Kruskal [Kruskal Jr, 1956] et Boruvka [Borůvka, 1926]. Nous aborderons ici les deux premières méthodes, et nous choisirons par la suite de nous focaliser sur l'algorithme de Prim. Nous supposons sans perdre de généralités que le graphe est connexe, si ce n'est pas le cas il suffit d'appliquer l'algorithme sur chaque composante connexe.

3.3.2 Algorithme de Kruskal

L'algorithme de Kruskal se base sur une des propriétés de connexité de l'arbre couvrant. En effet, un arbre couvrant est une composante connexe ayant le nombre d'arêtes minimum. Pour que le nombre d'arêtes soit minimal et que les arêtes choisies minimisent également la fonction de poids total alors dans un premier temps les arêtes sont triées par ordre de poids croissant.

A chaque étape de l'algorithme, l'arête de coût minimum reliant deux sommets appartenant à deux composantes connexes différentes est ajoutée. Le critère de minimalité de l'arête garantit la minimalité de la fonction de coût de l'arbre couvrant final. De plus,

si deux sommets x et y appartiennent à la même composante connexe alors il existe un chemin les reliant, si on ajoute l'arête (x, y) alors un cycle est créé et ce n'est plus un arbre.

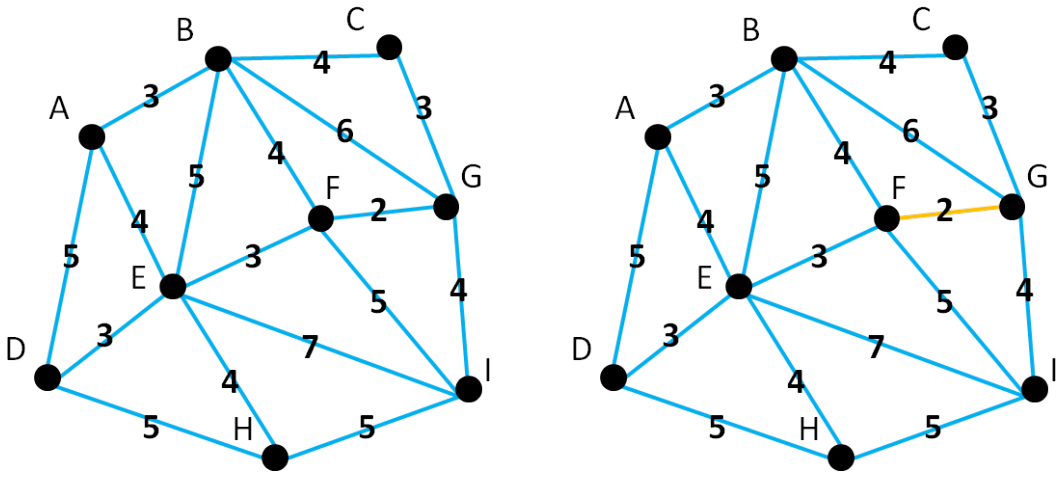
Lors de l'ajout de l'arête, la composante connexe est mise à jour. L'algorithme s'arrête lorsqu'il n'y a qu'une composante. Une des propriétés intéressantes de l'algorithme est qu'à tout instant le graphe construit est une forêt, c'est-à-dire que toutes les composantes connexes sont des arbres.

Entrées: Un graphe connexe $G = (V_G, E_G, \omega_G))$
Sorties: L'arbre couvrant de poids minimum $T = (V_T, E_T, \omega_G)$
initialiser $E_T = \emptyset$;
trier les arêtes e_i de G par $\omega_G(e_i)$ croissant;
tant que *T n'est pas connexe* **faire**
  ajouter à E_T l'arête de coût minimum reliant deux composantes connexes;
fin

Algorithm 1: Algorithme de Kruskal

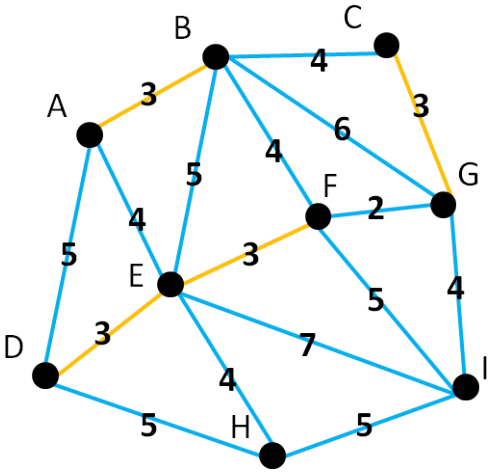
Nous allons dérouler l'algorithme en image sur un exemple. Sur cette série de schémas, les arêtes sont colorées en bleu, les sommets non-visités sont en en noir, les sommets ajoutés à l'arbre sont en rouge, les arêtes candidates pour être ajoutées à l'arbre sont en orange, celles qui sont ajoutées à l'arbre sont en rouge et enfin celles qui ne seront plus candidates seront grisées.

L'algorithme est en deux phases, la première consiste à créer une liste d'arêtes ordonnée et la deuxième est consacrée à la construction de l'arbre.

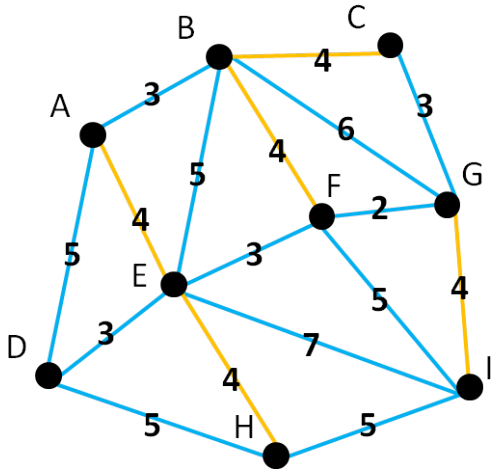


(i) (ii)

(i) Le graphe est non-orienté, et valué. Les poids des arêtes sont des valeurs positives et non nulles. Pour des raisons de commodité nous avons nommé les sommets de A à I, et le poids des arêtes sont des valeurs entières de 2 à 7. (ii) On note L la liste des arêtes ordonnée, à laquelle nous ajoutons celles qui ont la plus petite valeur. On a donc : $L = (\{F, G\})$.

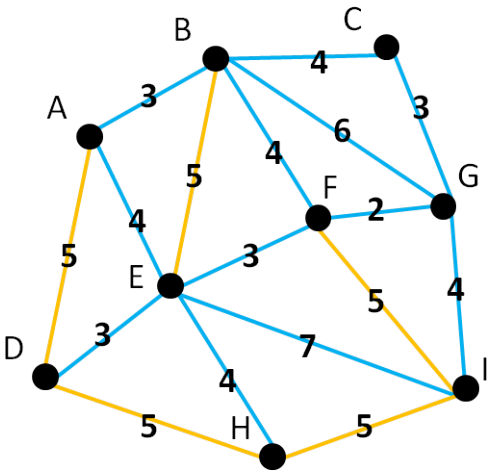


(iii)

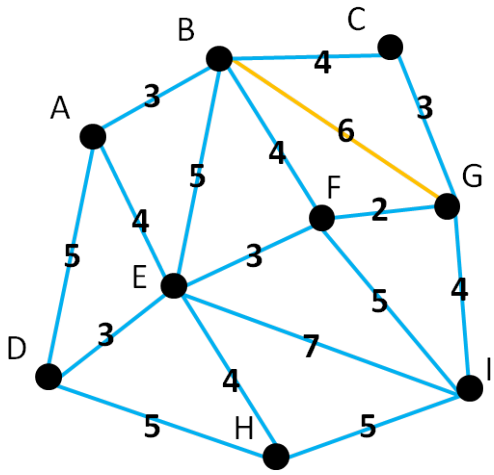


(iv)

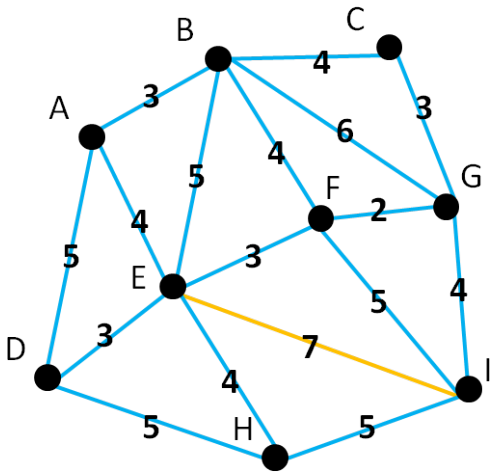
(iii) On ajoute à L les arêtes de valeurs 3. Le problème est qu'il y a plusieurs arêtes. Pour l'algorithme l'ordre que nous donnerons aux arêtes importe peu, à la fin nous aurons un arbre couvrant de poids minimum. Mais suivant l'ordre dans lequel les arêtes sont rangées, le résultats peut être différent. Pour l'exemple, nous avons nommé les sommets et nous rangerons les arêtes dans l'ordre lexicographique. Ce qui donne : $L = (\{F, G\}, \{A, B\}, \{C, G\}, \{D, E\}, \{E, F\})$. (iv) Nous itérons le processus avec les arêtes de poids 4 ; $L = (\{F, G\}, \{A, B\}, \{C, G\}, \{D, E\}, \{E, F\}, \{A, E\}, \{B, C\}, \{B, F\}, \{E, H\}, \{G, I\})$.



(v)



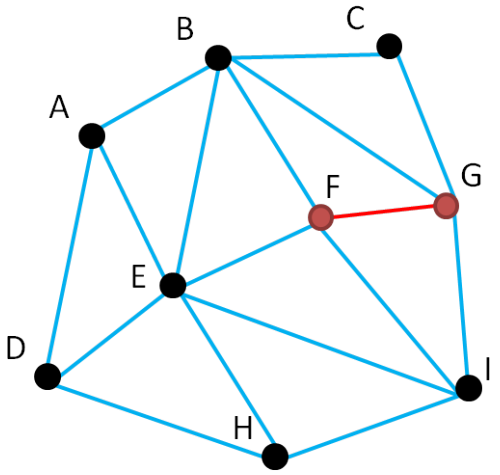
(vi)



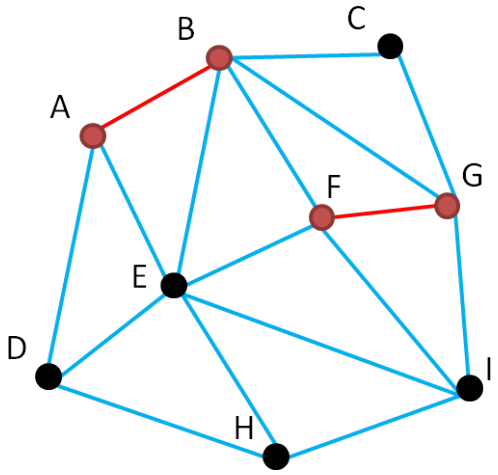
(vii)

On s'arrête lorsque toutes les arêtes ont été parcourues. On a ainsi : $L = (\{F, G\}, \{A, B\}, \{C, G\}, \{D, E\}, \{E, F\}, \{A, E\}, \{B, C\}, \{B, F\}, \{E, H\}, \{G, I\}, \{A, D\}, \{B, E\}, \{D, H\}, \{F, I\}, \{H, I\}, \{B, G\}, \{E, I\})$.

Nous en venons à la deuxième phase : la construction de l'arbre.

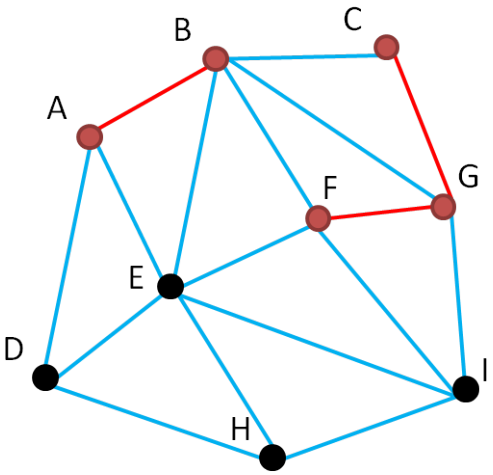


(viii)

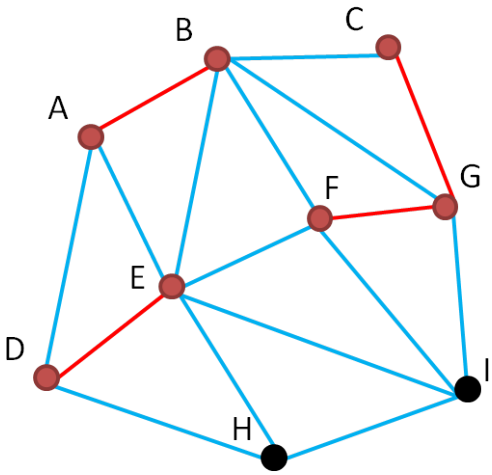


(ix)

(viii) On prend la première arête de la liste $\{F, G\}$ et on l'ajoute à l'arbre. (ix) Puis on ajoute la deuxième $\{A, B\}$. On remarque que les deux arêtes ne sont pas reliées. Le graphe construit n'est pas un arbre, mais ses composantes connexes sont des arbres.

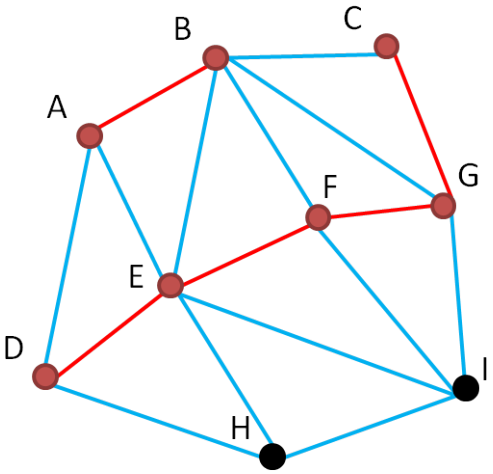


(x)

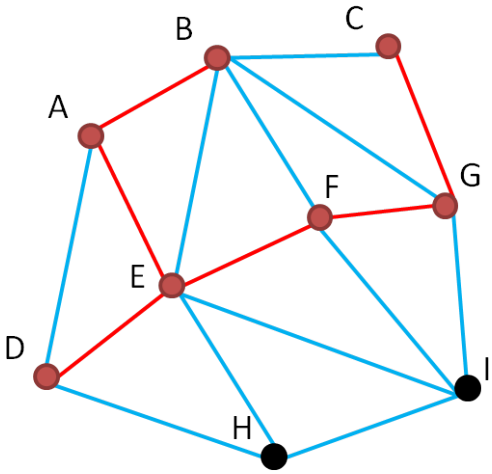


(xi)

(x) Continuons avec la troisième arête $\{C, G\}$. A partir de cette étape, il faut vérifier que l'ajout d'une arête ne crée pas de cycle dans la construction. Ici ce n'est pas le cas, $\{C, G\}$ est relié à $\{F, G\}$ sans créer de cycle. Le graphe est toujours une forêt avec deux composantes connexes (A, B) et (C, F, G) qui sont des arbres. (xi) La quatrième arête est $\{D, E\}$, on peut l'ajouter sans créer de cycle. Notre graphe se retrouve maintenant avec 3 composantes connexes.

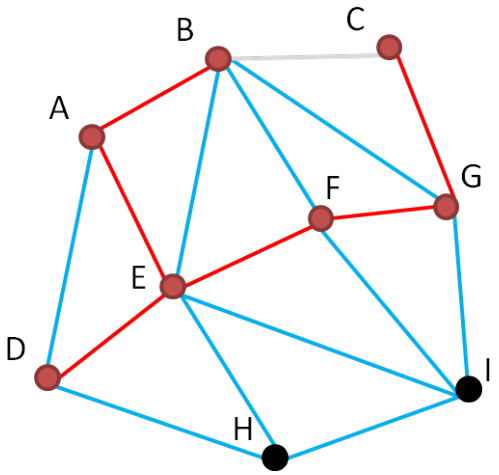


(xii)

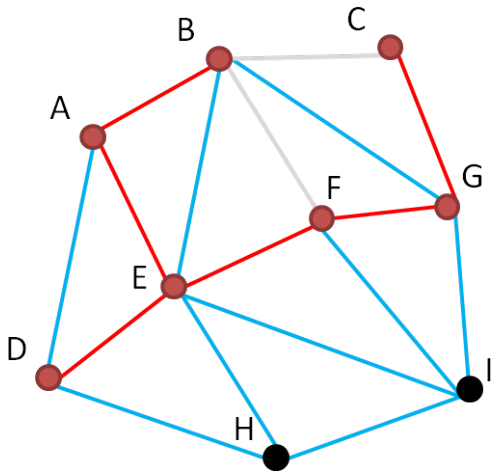


(xiii)

(xii) La cinquième arête $\{E, F\}$ relie deux composantes connexes sans créer de cycle. On peut encore l'ajouter sans problème à la construction. (xiii) De même, avec $\{A, E\}$. Notons que cette évolution à voulu que maintenant nous n'ayons qu'une composante connexe, et que cette composante est toujours un arbre.

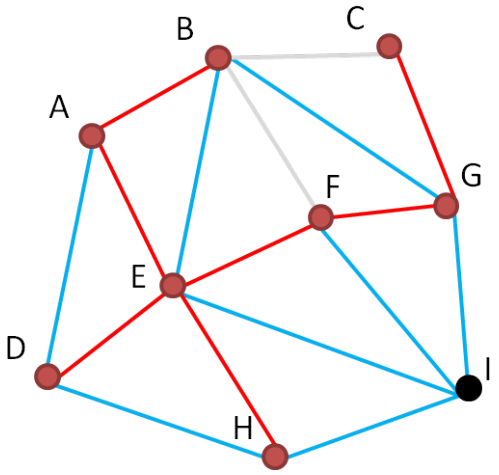


(xiv)

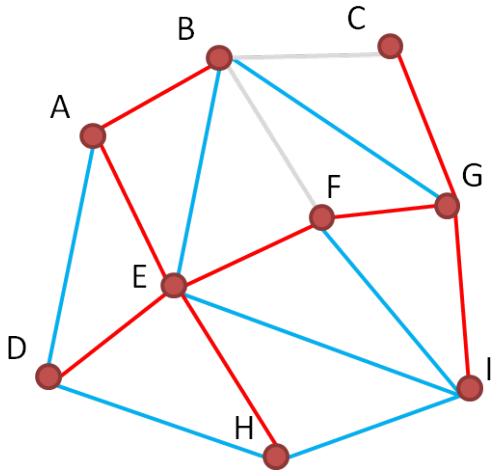


(xv)

(xiv) Il aura fallu attendre la septième arête pour voir un cas où nous ne pouvons pas ajouter cette arête à la construction. En effet, en ajoutant $\{B, C\}$ on créerait un cycle $\{A, B, C, G, F, E\}$ de longueur 6. Or un arbre n'a pas de cycle. Nous ne pouvons pas ajouter $\{B, C\}$. (xv) Même justification avec $\{B, F\}$, $\{A, B, F, E\}$ serait un cycle de longueur 4.



(xvi)



(xvii)

(xvi & xvii) Nous finissons l'algorithme par l'ajout de la neuvième et dixième arête qui ne pose aucune contrainte. Tous les sommets ont été ajoutés, l'algorithme s'arête.

3.3.3 Algorithme de Prim

L'algorithme de Prim repose sur la programmation dynamique. Le problème de l'arbre couvrant de poids minimum est un problème d'optimisation où la fonction de poids de l'arbre couvrant doit être minimisé. Le principe de la programmation dyna-

mique repose sur le fait que toute solution optimale s'appuie sur un sous-problème résolu localement de façon optimale.

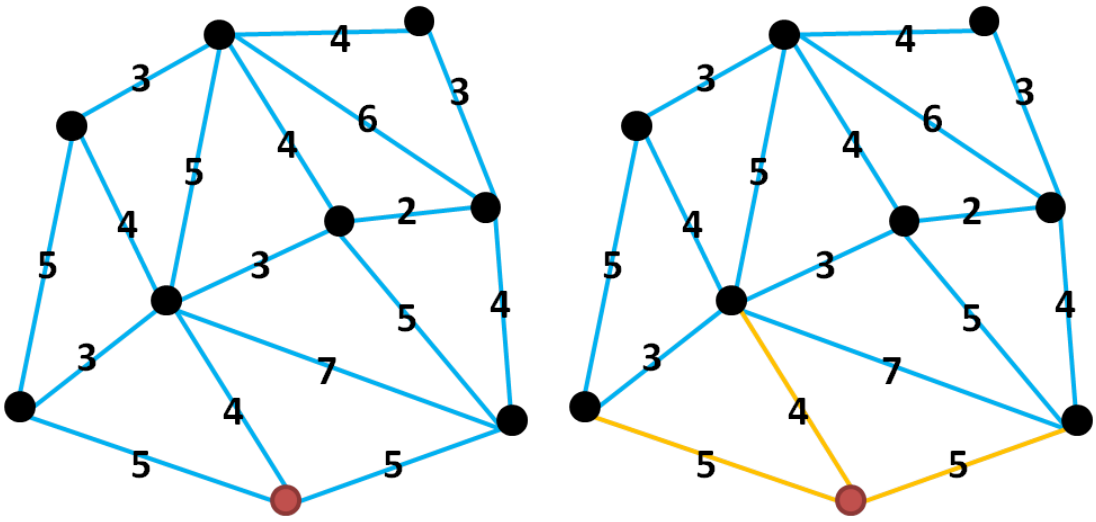
A chaque étape de l'algorithme, on ajoute un point à la construction de l'arbre. Supposons que nous sommes à l'étape i de l'algorithme, nous avons construit un arbre avec $i+1$ sommet d'un graphe. A l'étape $i+1$, le sommet ajouté n'influe pas sur le calcul précédent, sinon il aurait été préférable de le choisir à une étape antérieure pour améliorer la solution.

La construction de l'arbre se fait de manière gloutonne, à chaque étape de l'algorithme nous maintenons un arbre jusqu'à parcourir tous les sommets.

Entrées: Un graphe connexe $G = (V_G, E_G, \omega_G))$
Sorties: L'arbre couvrant de poids minimum $T = (V_T, E_T, \omega_G)$
initialiser $E_T = \emptyset$ et $V_T = \emptyset$;
sélectionner un point racine v_1 et l'ajouter à V_T;
répéter
    selection le sommet $v_i \notin V_T$ le plus proche de V_T;
    ajouter l'arête e_i à E_T connectant v_i à V_T;
    ajouter le sommet v_i à V_T;
jusqu'à *recouvrir tous les sommets de* G;

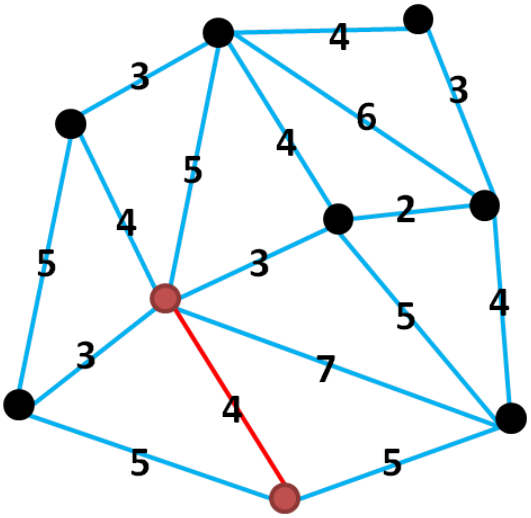
Algorithm 2: Algorithme de Prim

Comme précédemment, nous allons montrer l'évolution de l'algorithme en image sur un exemple avec le même code couleur.

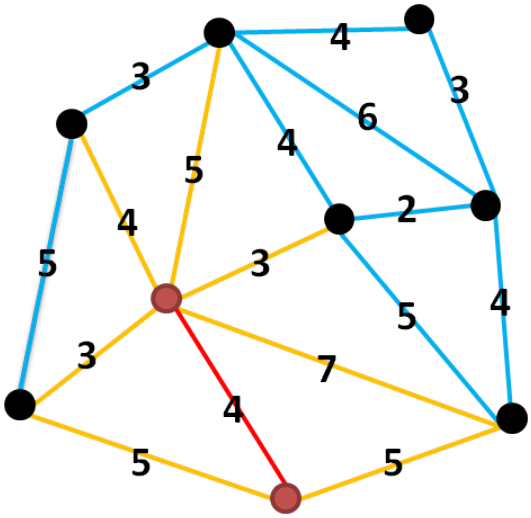


(i) (ii)

(i) Le graphe est non-orienté, et valué. Les poids des arêtes sont des valeurs positives et non nulles. Nous choisissons un sommet qui sera la racine de l'arbre et le point de départ de l'algorithme. (ii) A partir de la racine, nous sélectionnons les arêtes incidentes qui sont candidates pour être ajoutées à l'arbre.

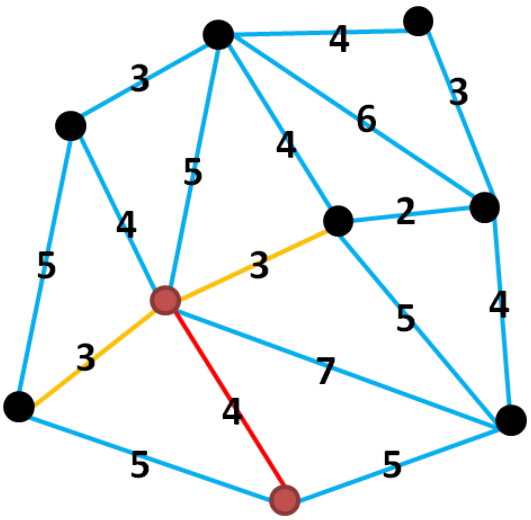


(iii)

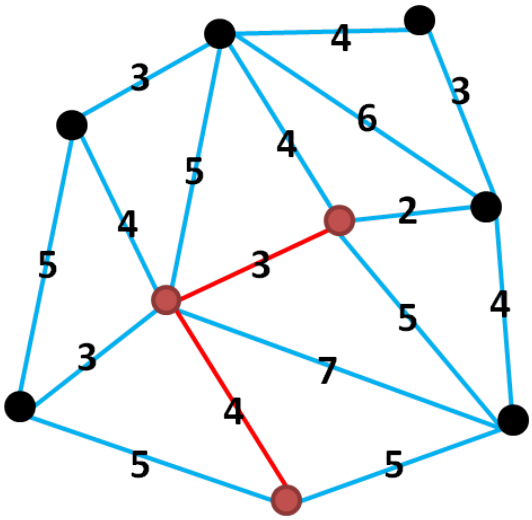


(iv)

(iii) L'arête de poids minimum est ajoutée à l'arbre. A cette étape l'arbre est composée d'une arête et de deux sommets. (iv) Comme à l'étape ii. de l'algorithme, nous sélectionnons l'ensemble des arêtes incidentes aux sommets de l'arbre qui sont candidates.

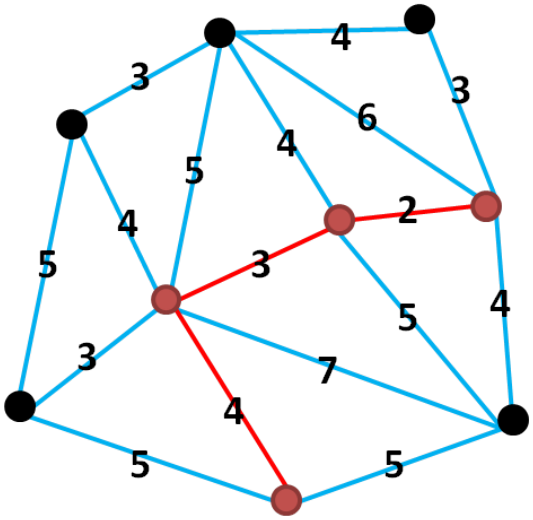


(v)

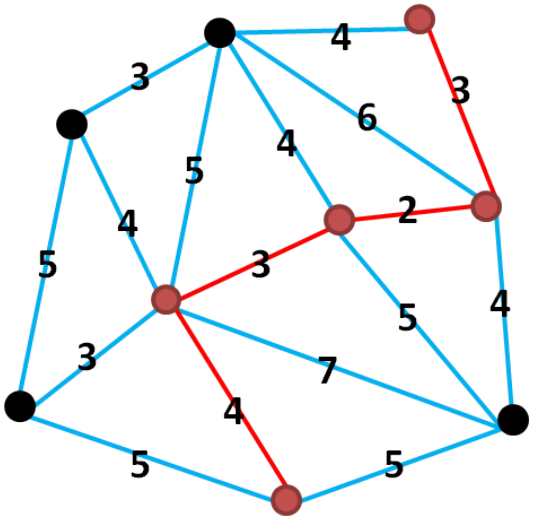


(vi)

(v) Cependant dans ce cas, nous avons deux arêtes dont le poids est minimum sur l'ensemble des arêtes candidates. (vi) Nous devons en choisir une. L'algorithme ne nous précise pas laquelle choisir, mais nous verrons pas la suite que nous serons obligés de forcer le choix de manière à avoir un algorithme déterministe qui donne toujours le même résultat.

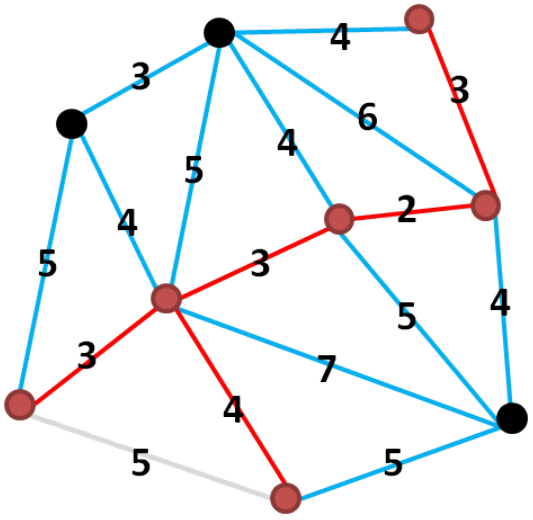


(vii)

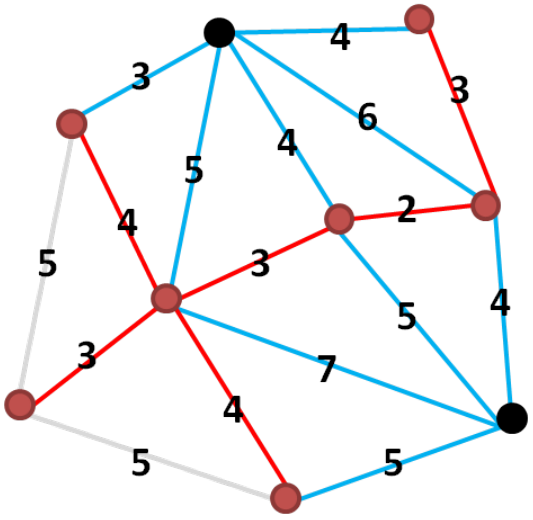


(viii)

(vii) Nous poursuivons le développement de l'algorithme, à cette étape ce n'est plus une arête de poids 3 qui est minimum, mais de poids 2 ; il n'y a pas de problème lors du choix de l'arête à ajouter à l'arbre. (viii) Ici, nous retombons sur le cas de l'étape vi. où nous devons choisir une arête.

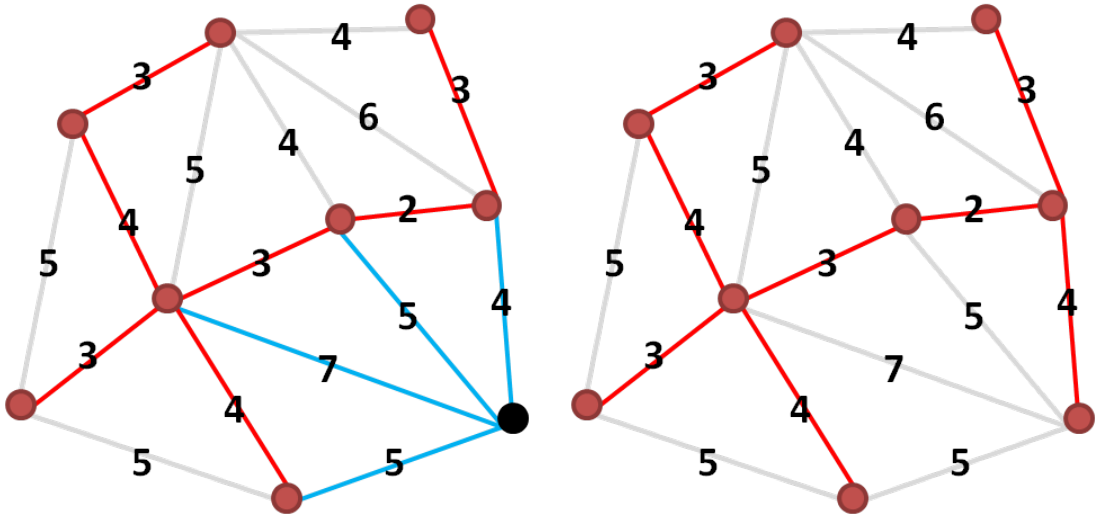


(ix)



(x)

(ix) Lors de l'ajout de cette arête, nous illustrons par l'arête grisée, quelque chose que l'algorithme ne prend pas en compte. A partir de maintenant, l'arête grisée sera toujours sélectionnée comme étant candidate pour être ajoutée à l'arbre. L'algorithme garantit qu'elle ne le sera jamais, car l'arête que l'on ajoute doit relier un sommet qui à été visité à un sommet qui ne l'a pas encore été.



(xi) (xii)

(xii) L'algorithme se poursuit, jusqu'à recouvrir tous les sommets. Tous les sommets ont été visités.

3.4 Sensibilité des arbres couvrants de poids minimum

En lien avec le problème de synchronisation dans le tatouage numérique, pour des critère de robustesse, nous avons besoin de connaître la stabilité de la structure que nous utilisons. Dans cette section, nous nous intéressons au problème de sensibilité des ACPM. Il peut être abordé de plusieurs façons différentes. Nous en présenterons quatre. Tout d'abord nous allons définir le problème d'une façon et explorer les possibilités d'aborder le problème.

Pour un graphe $G = (V_G, E_G, \omega_G)$ et $T = (V_G, E_T, \omega_G)$ son arbre couvrant de poids minimum, le problème de la sensibilité est de déterminer pour chaque arête $e_i \in E_G$ quelles valeurs de poids $\omega(e_i)$ sont possibles sans affecter la minimalité de l'arbre couvrant.

En d'autres mots on cherche pour chaque arête un intervalle $I_i =]\omega_G(e_i)^-; \omega_G(e_i)^+[$ tel que $\forall e_i \in E_G$, si pour une fonction de poids ω^*_G, $\omega^*_G(e_i) \in I_i$ alors l'arbre couvrant de poids minimum $T^* = (V_G, E_{T^*}, \omega^*_G)$ de G n'est pas modifié.

Ce problème est classé comme étant polynomial dans le cas des ACPM. Nous allons voir quatre façons d'aborder le problème : tout d'abord de manière globale, en calculant la puissance de la perturbation maximale qui ne modifie pas les connexions de l'arbre, de manière locale en calculant un intervalle de valeur possible ou en estimant la robustesse des différentes connexions et par une reformulation du problème de cinétique des graphes.

3.4.1 Sensibilité globale des arbres couvrants de poids minimum

Contrairement à l'approche décrite précédemment qui est un calcul local, dans [Gordeev, 2001] l'approche est globale. Avec les notations précédentes, nous avons $G = (V_G, E_G)$ un graphe, $T = (V_G, E_T) \in \mathcal{T}_G$ un arbre couvrant de G avec $E_T \subset E_G$ et $\mathcal{T}_G$, l'ensemble des arbres couvrants de G. Posons $A = (a_i = \omega(e_i))_{e_i \in E_T}$, le vecteur des valeurs attribuées aux arêtes du graphe G et qui appartiennent à l'arbre T. Ainsi le poids de l'arbre, correspondant à la somme des poids de toutes les arêtes le composant, s'exprime de la façon suivante :

$$\omega(T) = \sum_i a_i$$

Le problème de la recherche d'un ACPM est un un problème d'optimisation discrète noté par le triplet $(E, \mathcal{T}_G, A)$

3.4.2 Sensibilité locale des arbres couvrants de poids minimum

Dans le cas de la sensibilité des ACPM, il a été prouvé [Tarjan, 1982] que l'algorithme est polynomial et de complexité $o(m \cdot (m, n))$ avec $m = |E|$ et $n = |V|$.

Pour chaque arête, l'approche de [Dixon *et al.*, 1992] consiste à diviser le problème en deux parties. Tout d'abord, les auteurs se demandent pour les arêtes qui appartiennent à l'ACPM quelle est la quantité qu'ils peuvent ajouter aux arêtes afin qu'elle reste dans l'ACPM. D'autre part, pour les arêtes du graphe qui n'appartiennent pas à l'ACPM, on applique le procédé inverse. Dans ce cas, les auteurs se demandent quelle quantité peut être retirée de manière à ce que ces arêtes ne soient toujours pas sélectionnées dans l'ACPM.

On notera r_i^+ la quantité qu'on peut rajouter à l'arête e_i et respectivement r_i^- la quantité qu'il est possible de retirer à l'arête e_i.

3.4.3 Graphes dynamiques

Nous allons introduire ici un problème différent, qui nous servira à montrer une autre façon d'exprimer la sensibilité des ACPM.

Nous avons à disposition un graphe G euclidien dans un espace de dimension d. Nous nous intéressons à l'évolution de son arbre couvrant de poids minimum euclidien lorsqu'on déplace linéairement un certain nombre de points. Sous l'hypothèse que les points se déplacent à vitesse constante le long d'une droite [Katoh *et al.*, 1995], l'objectif est d'estimer des bornes minimale et maximale du nombre de changements de connexions dans l'arbre lorsque les points sont déplacés (figure 3.4). Nous ne nous attarderons

pas sur les résultats mais sur les hypothèses d'études et les conditions de ce problème [Rahmati et Zarei, 2010; Monma et Suri, 1992].

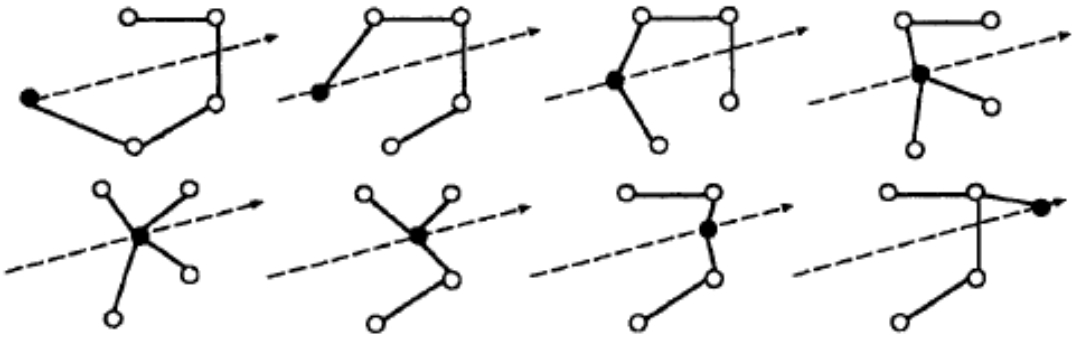


FIGURE 3.4 – Représentation de l'ensemble des arbres couvrants de poids minimum en fonction du déplacement d'un point le long d'une droite.

Avec n points, dans un graphe complet il y a $o(n^2)$ arêtes qui le composent, donc $o(n^2)$ distances. Dans le cas des transitions, naïvement la combinatoire est de l'ordre de $o(n^4)$. En d'autres mots l'objectif est d'approcher la réelle valeur du nombre de transitions.

Reprenons notre problème de sensibilité des ACPME, et mettons nous dans les mêmes conditions que ce problème. Nous sélectionnons un point que l'on déplace à vitesse constante le long d'une ligne droite. Ce qui nous intéresse est le cas limite où nous conservons les mêmes connexions dans l'arbre. Le problème de dynamique des graphes dans les ACPME peut être vu comme le "dual", c'est-à-dire lancer la simulation en déplaçant les points et s'arrêter avant qu'une transition soit effective dans l'arbre.

3.4.4 Robustesse des arêtes d'un ACPM

Dans cette approche de ACPM robuste [Salazar-Neumann, 2007], les auteurs considèrent tous les arbres couvrant possibles $T(G)$ d'un graphe G. Pour une valuation sur les arêtes ω_G, l'arbre couvrant de poids minimum est l'arbre $T \in T(G)$ qui minimise $w_G(T)$.

A partir d'une valuation ω_G, les auteurs perturbent ces valeurs. Leur objectif est de savoir quelles sont les conséquences de leur perturbation sur le poids de l'ACPM et d'en déduire des propriétés de robustesse sur les arêtes. A partir de ces notions, et en fonction de la perturbation, il est théoriquement possible de savoir si l'arête est robuste, c'est-à-dire qu'elle se retrouvera dans tous les ACPM quelle que soit la perturbation appliquée sur le graphe.

Notons que ce problème est bien plus général que ce que nous étudions. Il fait partie des problèmes $\mathcal{NP}$-difficiles notamment à cause de la recherche des arbres couvrants qui est exponentielle.

3.5 Conclusion

Dans cette section nous avons présenté la problématique des arbres couvrants de poids minimum en s'attardant sur la construction de ces structures par les algorithmes de Prim et de Kruskal. Ces algorithmes nous posent des problèmes d'unicité de l'arbre, une propriété essentielle lors d'une phase de synchronisation. Par la suite nous détaillerons nos choix en matière d'algorithme et les solutions que nous avons mises en place pour résoudre ce problème.

Par ailleurs, nous avons présenté un état de l'art complet sur la sensibilité des arbres couvrants de poids minimum. Chaque méthode nous ont inspiré pour concevoir une étude complète de la sensibilité des sommets dans un arbre couvrant de poids minimum euclidien.

Deuxième partie

Contributions

Chapitre 4

Sensibilité des ACPME

Préambule

Afin d'utiliser la structure des arbres couvrants de poids minimum euclidien (ACPME) pour la synchronisation dans les maillages 3D, nous proposons ici d'analyser sa sensibilité. Comme nous l'avons vu dans le chapitre précédent, l'étude de la sensibilité des APCM a été réalisée de différentes manières. Généralement les études proposées se basent sur des graphes quelconques. Nous verrons ici quelles sont les spécificités de notre problème.

Sommaire

4.1 Introduction

Dans le cas de données géométriques, l'évolution du graphe est différente. Tout d'abord, le graphe est complet. C'est à dire qu'il existe une arête entre deux sommets

différents. Ce graphe est valué généralement par une distance, par défaut nous prendrons la distance euclidienne.

De plus, géométriquement le changement de valeur d'une arête se traduit par un déplacement d'un point par rapport aux autres. Ainsi, le déplacement d'un point affecte les distances avec tous les autres ce qui entraîne une modification de toutes les valeurs des arêtes du graphe incidentes au point déplacé.

C'est pourquoi nous proposons de constater la fragilité des ACPME puis de proposer une méthode d'approximation permettant d'évaluer la mobilité d'un point dans un graphe de telle façon à ce que la structure ne soit pas modifiée. Nous ferons des hypothèses sur le déplacement des points.

Mais dans un premier temps nous allons nous intéresser à la construction de l'arbre. Nous devons choisir entre les deux algorithmes présentés précédemment : l'algorithme de Prim et celui de Kruskal. De plus nous avons vu que l'arbre couvrant de poids minimum n'est pas unique, ce qui est un problème dans le cas de la synchronisation. Nous allons voir comment remédier à ce problème en rendant un résultat unique par un algorithme totalement déterministe.

4.2 Choix de l'algorithme

Nous avons choisi l'algorithme de Prim pour la construction des ACPME, nous allons expliquer ici les avantages de ce dernier par rapport à celui de Kruskal pour notre application.

Les algorithmes sont similaires, mais comme nous l'avons vu précédemment chacun propose une approche différente. La différence se fait dans le choix de l'invariant. Pour Kruskal à toute étape nous avons une forêt qui est incluse dans l'ACPM final. Quant à Prim, on construit étape par étape de manière gloutonne ce qui deviendra le futur ACPM.

On peut considérer que Kruskal a une approche plus globale du graphe. Sa première étape étant un tri des arêtes du graphe, peu importe où elles se trouvent les unes par rapport aux autres, l'ordre de parcours est fixé par ce tri. Prim a une approche plus locale, une fois l'arbre construit, il s'interroge sur l'arête qui sera ajoutée en fonction de la construction en cours. L'ordre des arêtes sélectionnées dépend du point du départ et de l'évolution de la construction.

C'est clairement cet argument qui fait que nous avons choisi Prim. Pour être plus précis et s'en convaincre, il suffit de prendre l'exemple que nous avons montré lors de la présentation de ces algorithmes et de s'intéresser au cas où nous avons des arêtes qui ont la même valeur.

En regardant la figure 4.1, nous sommes à une étape de l'algorithme où nous avons le choix entre deux arêtes pour l'intégrer dans l'arbre. En fonction du choix le résultat final

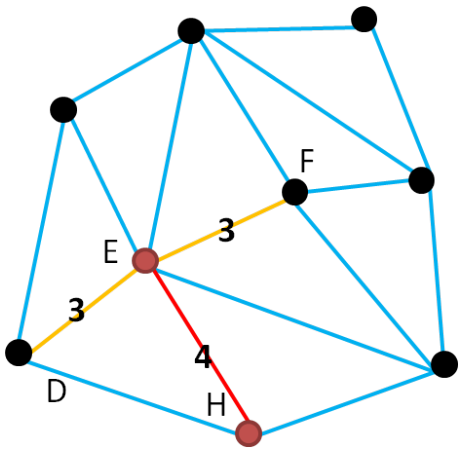


FIGURE 4.1 – Illustration d'un graphe à une étape de l'algorithme de Prim. Les arêtes bleues représentent le graphe, en rouge l'arbre couvrant de poids minimum en construction et en orange les arêtes candidates pour intégrer l'ACPM.

peut être différent. L'algorithme ne se préoccupe pas des choix faits par l'utilisateur, son objectif est seulement de rendre un résultat parmi les résultats possibles.

Pour en revenir à Kruskal, cette notion de choix est moins visible. Tout se passe au niveau du tri des arêtes. Une fois le tri réalisé l'algorithme se déroule sans se poser de question sur l'existence d'autres solutions possibles. Pourtant si j'ai deux arêtes e_1 et e_2 de valeurs égales, laquelle sera la première dans la liste ordonnée ? Est-ce qu'il y a une incidence sur le résultat final d'en privilégier une par rapport à l'autre ? Nous ne savons pas, mais comme sur l'algorithme de Prim l'objectif n'est que de rendre un résultat parmi ceux qui sont possibles.

Puisqu'avec l'algorithme de Prim nous avons le choix, et qu'il est explicite, nous avons donc la possibilité de rendre un résultat et toujours le même du moment que le sommet initial choisi est identique. C'est ce que nous décrivons dans la section suivante.

4.3 Unicité de l'arbre

Toujours en nous basant sur la figure 4.1, l'objectif ici est de décider laquelle des deux arêtes (DE ou EF) nous allons choisir.

Tout d'abord, nous reformulons le problème. On ne cherche pas à savoir quelle arête sera sélectionnée, mais quel sera le point sélectionné. Dans l'exemple, il s'agira de savoir qui de D ou de F sera choisi.

Résoudre ce problème, c'est équivalent à imposer une priorité d'un point sur un autre, c'est dire lequel sera le premier suivant un certain ordre. Nous retombons sur un

problème d'ordre, de relation entre les points qui sont les problématiques récurrentes de la synchronisation.

Une solution serait de prendre l'ordre lexicographique, dont la définition est donnée ci-dessous. Suivant la relation $<$, pour $a < b$ nous dirons par habitude que "a est plus petit que b". Cette relation est transitive. Pour rappel, si $a < b$ et $b < c$ alors $a < c$. Ainsi, s'il y a un choix à faire entre deux ou plusieurs points, il suffit de prendre le plus petit pour cette relation par exemple.

$$a, b \in \mathbb{R}^3 : a < b \Longleftrightarrow (a_x < b_x) \vee (a_x = b_x \wedge a_y < b_y) \vee (a_x = b_x \wedge a_y = b_y \wedge a_z < b_z)$$

Cependant, si on conserve le système de coordonnées cartésiennes, la relation d'ordre devient sensible à certaines opérations comme par exemple la rotation. Il faut dont que le repère de référence soit lié à l'objet.

4.4 Constat de la fragilité des ACPME

Dans un premier temps, nous analysons expérimentalement la sensibilité d'un ACPME. Prenons un maillage $\mathcal{M}$, le graphe complet $G = (V_G, E_G, \omega_G)$ et calculons $T = (V_G, E_T, \omega_G)$ son unique ACPME. Pour analyser la sensibilité de la structure, nous bruitons légèrement l'ensemble des points par un bruit gaussien suivant la loi normale $\mathcal{N}(\sigma, \mu)$ sur les 3 coordonnées cartésiennes.

Nous notons $\mathcal{M}_\sigma$ le maillage bruité et $G_\sigma = (V_G, E_G, \omega_\sigma)$ le nouveau graphe correspondant. Par l'algorithme de Prim, calculons $T_\sigma = (V_G, E_{T_\sigma}, \omega_\sigma)$. Ici, une autre problématique se pose, il faut évaluer la vraisemblance entre les deux arbres T et T_σ.

Dans le cas général, le problème de comparaison des arbres est $\mathcal{NP} - complet$. Il n'existe donc pas de solution calculable en un temps polynomial. Nous passons donc par une heuristique qui repose sur la connaissance du maillage $\mathcal{M}$ et la position des points.

Nous connaissons la position des points initiaux, leur déplacement, ainsi que leurs connexions avec les autres sommets. Nous choisirons comme mesure de vraisemblance le taux de connexions communes entre l'arbre du maillage bruité et celui du maillage original. Pour cela, nous cherchons parmi les connexions qui sont dans l'arbre d'origine, celles qui sont conservées après l'ajout du bruit. Notons $\mu(T, T_\sigma)$ la mesure de vraisemblance entre les arbres T et T_σ, nous obtenons alors formellement :

$$\mu(T, T_\sigma) = \frac{1}{m} \sum_{e_i \in E_T} \delta_{\{e_i \in E_{T_\sigma}\}}. \tag{4.1}$$

Nous prenons trois objets quelconques sur lesquels nous appliquons un bruit gaussien sur chacune des coordonnées. Nous illustrons sur la figure 4.2, les résultats sur ces trois maillages. Au cours de cette expérience le bruit est totalement quelconque et ne prend pas en compte les caractéritiques de l'objet, ni de correction sur l'algorithme de Prim.

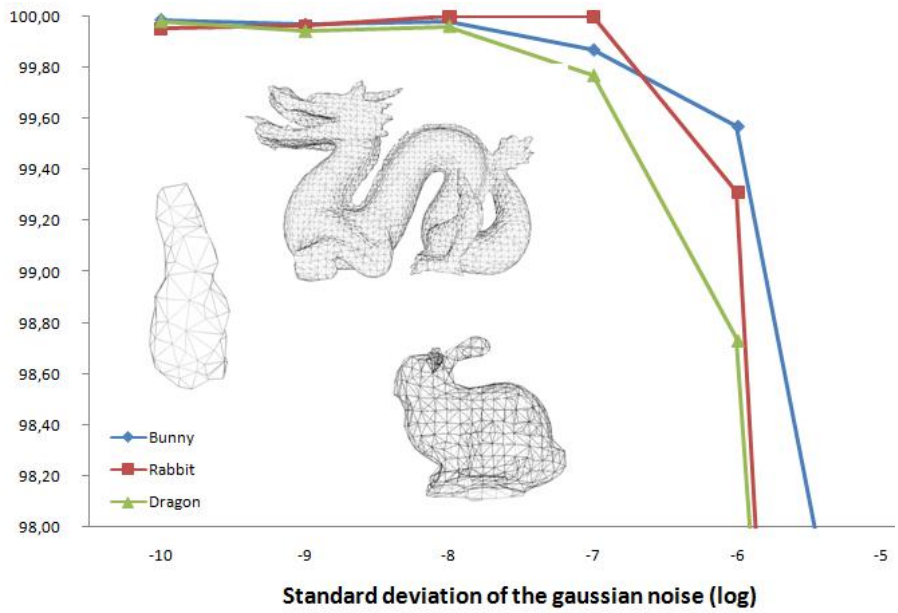


FIGURE 4.2 – Représentation graphique du nombre d'arêtes communes des ACPME en fonction de l'écart type d'un bruit Gaussien appliqué sur les trois coordonnées cartésiennes.

Cette expérience révèle deux choses. Tout d'abord, pour des valeurs de σ comprise entre 10^{-10} et 10^{-7}, l'impact sur le maillage est quasi nul étant donné que la précision des coordonnées est de l'ordre de 10^{-7}. Pourtant le résultat du calcul de vraisemblance n'est pas de 100%. Nous justifions ce résultat par le problème d'unicité de l'arbre.

Lorsque σ atteint 10^{-6}, la chute de la vraisemblance entre l'arbre original et l'arbre bruité chute considérablement. Ce qui montre ainsi la fragilité d'un arbre au déplacement des points.

Remarquant la sensibilité des ACPME pour des déplacements infimes, nous nous sommes donc intéressés à l'étude de cet objet. Essayons à partir de l'algorithme de Prim, d'évaluer une cellule dans laquelle le sommet peut se déplacer sans changer les connexions dans l'arbre. Le problème de l'étude de la sensibilité est $\mathcal{NP}-$complet, c'est pour cela que nous avons besoin de simplifier le problème.

4.5 Simplification du problème

Tout d'abord, nous réécrivons l'algorithme de Prim où nous mettons en évidence l'évolution de la construction de l'arbre au fil des étapes. Notons $T_i = (V_i, E_i, \omega_G)$, le

graphe construit à l'étape i de l'algorithme à partir du graphe $G = (V_G, E_G, \omega_G)$.

Entrées: Un graphe connexe $G = (V_G, E_G, \omega_G))$
Sorties: L'arbre couvrant de poids minimum $T = (V_T, E_T, \omega_G)$
initialiser $E_0 = \emptyset$, $V_0 = \emptyset$;
sélectionner un point racine v_1, $V_1 = \{v_1\}$;
répéter
  soit $v_{i+1} \in V_G \backslash V_i$, le sommet le plus proche de V_i ;
  soit $f(v_{i+1}) \in V_i$, le sommet qui connecte v_{i+1} à V_i ;
  $V_{i+1} = V_i \cup \{v_{i+1}\}$;
  $E_{i+1} = E_i \cup \{\{v_{i+1}, f(v_{i+1})\}\}$;
jusqu'à $V_{i+1} = V_G$;

Algorithm 3: Algorithme de Prim réécrit.

A chaque étape de l'algorithme nous ajoutons le sommet le plus proche v_{i+1} de l'ensemble de points V_i ainsi que l'arête $\{\{v_{i+1}, f(v_{i+1})\}\}$ au graphe T_i pour construire l'arbre T_{i+1}. Nous notons $f(v_{i+1})$ le père de v_{i+1}, le sommet qui connecte v_{i+1} à V_i lors de la construction de l'ACPME.

Avec ces notations, nous proposons une approximation du déplacement possible des points dans le maillage sans changer de connexion dans l'ACPME. Pour cela nous sommes obligés de faire des hypothèses importantes pour simplifier le problème et le rendre abordable.

— A l'étape i de l'algorithme de Prim, nous étudions le déplacement possible d'un seul point v_i en supposant que l'ensemble des sommets précédemment sélectionnés ($\{v_k : k < i\}$) ne se déplacent pas ;

— Nous ne prenons pas en compte les modifications que le déplacement pourrait entraîner ultérieurement dans la construction de l'arbre ;

Notons que ces conditions sont très restrictives et ne sont pas représentatives des conditions réelles. L'objectif est d'obtenir une approximation et une valeur numérique quantifiant le déplacement possible d'un point.

Après le déplacement du point, v_i devient v^* et nous voulons que v^* vérifie trois propriétés :

— v^* ne sera pas sélectionné à l'étape $k < i$;

— v^* sera sélectionné à l'étape $k \geq i$;

— La connexion entre $f(v_i)$ et v^* est conservée. En d'autre mots $f(v^*) = f(v_i)$.

Dans ces conditions nous étudierons le cas où nous limitons les déplacements du point à une droite définie par celle qui passe par v_i et $f(v_i)$ et le cas général sans contrainte de déplacement.

4.6 Déplacement 1D dans les ACPME

Comme nous l'avons vu dans la section 3.4, le problème de sensibilité des ACPM peut être abordé de différentes manières. Pour notre méthode, nous nous sommes principalement inspirés des méthodes de calcul local [Dixon *et al.*, 1992] et de cinétique des graphes [Salazar-Neumann, 2007].

En effet, en faisant le parallèle avec la cinétique des graphes, nous ne nous intéressons pas au nombre de transitions que l'ACPME subit lors du déplacement du point mais du problème trivial. C'est-à-dire, que nous souhaitons connaître les limites du déplacement du point tant que la structure est conservée. Comme dans la plupart de ces problèmes, on considère que le point se déplace le long d'une droite. Celle-ci est déterminée par la position des points v_i et $f(v_i)$.

Nous allons sur cette droite calculer un intervalle, donc deux valeurs limites correspondant à la distance à laquelle le point peut être rapproché de son père r^- (4.6.2) et la distance d'éloignement r^+ (4.6.1).

4.6.1 Calcul de la distance d'éloignement.

L'algorithme de Prim sélectionne à chaque étape i le sommet v_{i+1} le plus proche de l'ensemble de points courant V_i. L'une des premières limites, simple à calculer et à justifier, est la distance entre V_i et le deuxième point le plus proche. Nous notons $s(v_i) \in V \backslash V_i$ ce point, et formellement nous obtenons la relation suivante :

$$\omega(f(v_i), v_i) < \omega((f \circ s)(v_i), s(v_i)).$$

De manière géométrique, cette condition ne suffit pas. En effet, en s'éloignant de $f(v_i)$, on peut se rapprocher d'un autre point qui appartient à l'ensemble V_i et ainsi changer le point de connexion de v_i à V_i.

Ce phénomène est illustré sur la figure 4.4. Dans le cas suivant, nous nous plaçons à l'étape de l'algorithme où l'arête $\{v_5, v_6\}$ est ajoutée à l'arbre. Déplaçons v_6 le long de la droite (v_5, v_6) en l'éloignant du sommet v_5. En éloignant trop le sommet de v_5, il se rapproche de v_2. Ainsi, en recalculant l'ACPME le résultat est différent.

La distance limite dépend des cellules de Voronoï 3D des sommets de V_i. La cellule de Voronoï d'un point $p \in V$ est l'ensemble des points les plus proches de p. Nous la notons $Vor(v_i)$, et nous avons :

$Vor_{V(v_i)} = \{x \in \mathbb{R}^3 : \forall v_j \in V d(x, v_i) < d(x, v_j)\}$

Ainsi la distance entre $f(v_i)$ l'intersection entre Δ_i et la cellule de Voronoï $Vor_{V_i(f(v_i))}$ donne une deuxième limite maximale de la distance $\omega(f(v_i), v_i)$, qu'on exprime de la façon suivante avec $x(v_i) \in \mathbb{R}^3 : x(v_i) = \Delta_i \cap Vor_{V_i(f(v_i))}$:

$$\omega(f(v_i), v_i) < d(f(v_i), x(v_i))$$

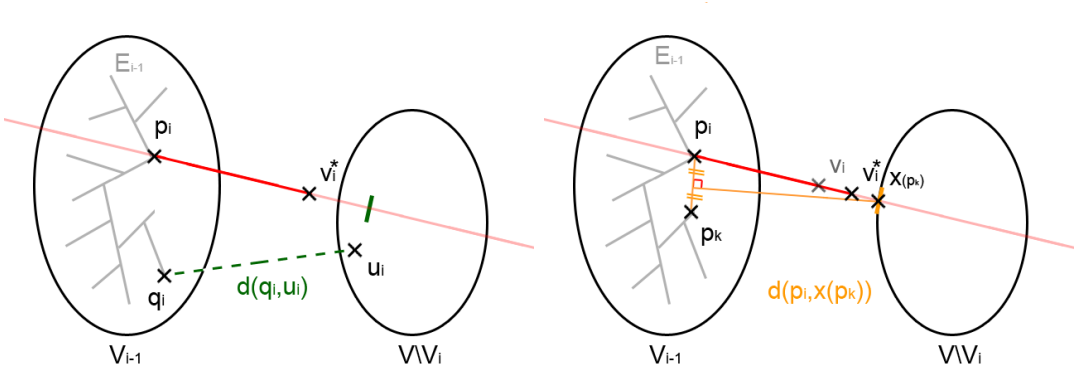


FIGURE 4.3 – Schématisation des deux calculs pour la distance d'éloignement. Sur la figure de gauche, le cas où nous prenons en compte la distance du deuxième sommet le plus proche à l'ensemble V_{i-1}. Et sur la droite, la représentation d'une intersection entre la droite de déplacement du point et la frontière des cellules de Voronoï des points p_i et p_k.

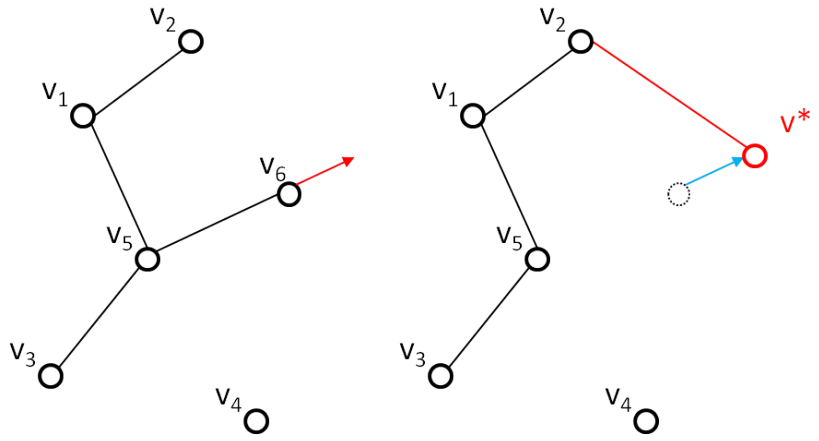


FIGURE 4.4 – Illustration du phénomène d'éloignement et d'un possible changement de parent dans la construction de l'arbre.

La distance d'éloignement doit vérifier les deux conditions décrites ci-dessus. Il suffit de prendre la valeur minimale, on a ainsi :

$$d_i^+ = \min\{\omega((f \circ s)(v_i), s(v_i)), d(f(v_i), x(v_i))\}.$$

Et on en déduit facilement, la distance d'éloignement possible :

$$r_i^+ = d_i^+ - \omega(\{f(v_i), v_i\}).$$

4.6.2 Calcul de la distance de rapprochement.

Pour rechercher la distance de rapprochement de v_i vers $f(v_i)$ nous nous servons de la relation fondamentale suivante :

$$\forall e_j \in E_{i-1} \backslash E_k; \omega(e_j) < \omega(\{f(v_i), v_i\}).$$

Cette relation signifie qu'entre l'étape k de l'algorithme où le sommet $f(v_i)$ a été sélectionné et l'étape courante, les arêtes sont toutes de poids plus faible que $\{f(v_i), v_i\}$. Comme nous nous intéressons aux cas limites de la valeur de $\omega(\{f(v_i), v_i\})$, alors d'après l'inégalité précédente on pose :

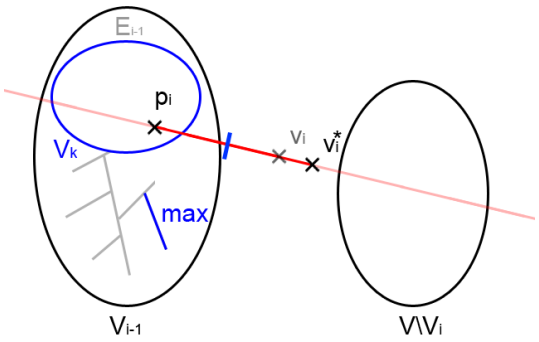


FIGURE 4.5 – Schématisation du calcul de la distance de rapprochement qui ne dépend que des distances précédemment calculées.

$$d_i^- = \max\{\omega(e_j) : e_j \in E_{i-1} \backslash E_k\}.$$

d_i^- est la distance minimum entre $f(v_i)$ et v_i (figure 4.5). Nous en déduisons facilement la distance de déplacement possible r_i^- :

$$r_i^- = \omega(\{f(v_i), v_i\}) - d_i^-.$$

4.6.3 Quantification du déplacement

Maintenant que nous avons des informations de déplacement avec r_i^+ et r_i^-, pour quantifier la mobilité du point nous prenons la distance minimum :

$$r_i = \min\{r_i^-, r_i^+\}.$$

On obtient donc un intervalle I_i dans lequel le sommet v^* pourrait se trouver (figure 4.6) :

$$I_i =]v_i - r_i \cdot f(v_i)v_i \ ; \ v_i + r_i \cdot f(v_i)v_i[$$

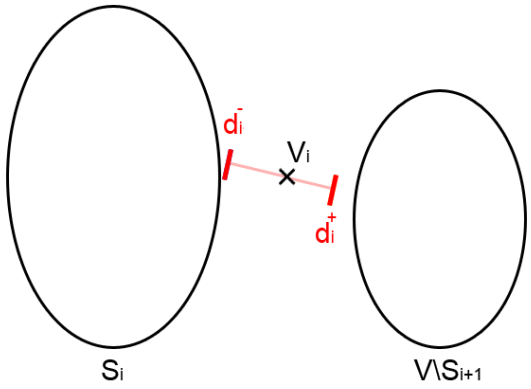


FIGURE 4.6 – Schématisation de l'intervalle de déplacement I_i, en fonction des valeurs calculées précédemment.

L'objectif est maintenant de prouver la validité expérimentale du résultat. Cette valeur doit être représentative de la mobilité du point. Avant cela, nous proposons dans la section suivante l'extension théorique à un déplacement du point dans les trois dimensions.

4.7 Déplacement 3D dans les ACPM

A la vue des résultats obtenus sur le déplacement restreint à une droite, il semblerait qu'on puisse étendre la propriété aux trois dimensions de l'espace. En effet, les résultats sur la distance d'éloignement sont indépendant de la droite choisie au déplacement du point. Avec les mêmes notations nous avons directement les résultats suivants :

$$v_i^* \in Vor_{V_i(f(v_i))} \cap B(f(v_i), \omega((f \circ s)(v_i), s(v_i)))$$

Pour simplifier les notations, nous notons cet espace A_i :

$$A_i = Vor_{V_i(f(v_i))} \cap B(f(v_i), \omega((f \circ s)(v_i), s(v_i)))$$

.

Au niveau de la distance de rapprochement, nous avons une distance limite entre $f(v_i)$ et v_i^* notée r_i^-. Si on applique cette restriction de rapprochement à l'ensemble des points de V_i nous obtenons un espace de déplacement qui serait interdit pour v_i^*. Nous notons F_i, cet espace et nous avons :

$$F_i = F_{i-1} \cup (\bigcup_{k=0}^{i-1} B(v_i, d_i^-)) \tag{4.2}$$

Ainsi de manière presque immédiate nous obtenons un espace de déplacement possible du point dans l'espace $A_i \backslash F_i$. La question est maintenant de savoir si nous pouvons affiner ce résultat, sinon nous allons le démontrer par construction.

Pour cela, à chaque étape i, dans la construction de l'EMST par l'algorithme de Prim, nous devons considérer l'espace dans lequel nous pouvons garantir qu'il n'y a aucun autre point que ceux qui ont déjà été ajoutés à l'arbre. Nous noterons cet espace C_i privé des sommets V_i, on a donc la propriété $C_i \cup V_i = V_i$ (i.e. $C_i = \emptyset$). Pour cela on procède par récurrence.

4.7.1 Démonstration par récurrence

Initialisation

$i = 0$: On choisit la racine $v_0 \in V \subset R^3$. Le choix de la racine est arbitraire, ainsi par convention on note l'espace du choix de la racine $C_0 = \emptyset$.

Première itération

$i = 1$: v_1 est le sommet le plus proche de v_0 d'après l'algorithme de Prim. Il n'existe donc aucun sommet dans la boule ouverte de centre v_0 et de rayon $l_2(v_0, v_1)$. On a :

$$C_1 = B_o(v_0, \omega(e_1)) \backslash \{v_0\}.$$

Deuxième itération

$i = 2$: v_2 est le sommet le plus proche de v_0 et v_1. On a deux cas possibles, v_2 est connecté soit à v_0 soit à v_1.

v_2 connecté à v_1. Comme précédemment il n'existe pas de sommet dans la boule ouverte de centre v_1 et de rayon $l_2(v_1, v_2)$, et également dans la boule ouverte de centre v_0 et de rayon $l_2(v_1, v_2)$, sinon un autre sommet aurait été sélectionné et serait connecté à v_0.

De plus, on a : $l_2(v_1, v_2) > l_2(v_0, v_1)$. Sachant que v_2 est le sommet le plus proche de v_0 et v_1, il n'y a pas de sommet dans la boule ouverte de centre v_1 et de rayon $l_2(v_0, v_2)$, sinon un autre sommet aurait été sélectionné et serait connecté à v_1.

On a donc l'ensemble suivant qui ne contient pas de sommet :
$C_{2,1} = C_1 \cup (B_o(v_0, l_2(v_0, v_1))\backslash\{v_0\} \cup B_o(v_1, l_2(v_0, v_1))\backslash\{v_1\})$

v_2 connecté à v_0. Toujours avec le même raisonnement, la boule ouverte de centre v_0 et de rayon $l_2(v_0, v_2)$ ne contient pas de sommet. De plus, on a : $l_2(v_1, v_2) > l_2(v_0, v_1)$. Sachant que v_2 est le sommet le plus proche de v_0 et v_1, il n'y a pas de sommet dans la boule ouverte de centre v_1 et de rayon$l_2(v_0, v_2)$, sinon un autre sommet aurait été sélectionné et serait connecté à v_1.

On en déduit l'ensemble suivant qui ne contient pas de sommet :
$C_{2,2} = C_1 \cup (B_o(v_0, l_2(v_0, v_2))\backslash\{v_0\} \cup B_o(v_1, l_2(v_0, v_2))\backslash\{v_1\})$

De manière générale on notera $f(v_i)$ le père du sommet v_i choisi $e_i = \{f(v_i), v_i\}$ la connexion de poids $\omega(e_i) = l_2(f(v_i), v_i)$ ajoutée dans l'arbre à l'étape i dans l'algorithme de Prim.

A partir de $C_{2,1}$, $C_{2,2}$ et avec les notations introduites, on en déduit C_2 :
$C_2 = C_1 \cup (B_o(v_0, \omega(e_2))\backslash\{v_0\} \cup B_o(v_1, \omega(e_2))\backslash\{v_1\})$

4.7.2 Relation de récurrence

On en déduit ainsi une relation de récurrence :

$$C_k = C_{k-1} \cup (\bigcup_{j=0}^{k-1} B_o(v_j, \omega(e_k))\backslash\{v_j\}) \tag{4.3}$$

Nous en déduisons que l'espace de déplacement possible du point v_i^* est l'espace $A_i\backslash C_i$. Il reste maintenant à faire le lien entre cette relation 4.3 et l'équation 4.2 que nous avons proposée en ce début de section.

4.7.3 Lien avec les résultats précédents

A partir de 4.3, faisons un simple jeu de réécriture. Nous allons rassembler toutes les boules qui ont le même centre. Ainsi, il suffira de choisir celle qui a le rayon le plus grand pour que cela simplifie notre écriture.

$$C_k = C_{k-1} \cup \left(\bigcup_{j=0}^{k-1} B_o(v_j, \omega(e_k)) \backslash \{v_j\} \right)$$

$$\Leftrightarrow C_k = C_{k-2} \cup \left(\bigcup_{j=0}^{k-2} B_o\left(v_j, \omega(e_k - 1)\right) \backslash \{v_j\} \right) \cup \left(\bigcup_{j=0}^{k-1} B_o\left(v_j, \omega(e_k)\right) \backslash \{v_j\} \right)$$

$$\Leftrightarrow C_k = \left(C_{k-2} \cup \left(\bigcup_{j=0}^{k-2} B_o\left(v_j, \omega(e_{k-1})\right) \cup B_o\left(v_j, \omega(e_k)\right) \right) \cup B_o\left(v_{k-1}, \omega(e_k)\right) \right) \backslash V_{k-1}$$

$$\Leftrightarrow C_k = \left(C_{k-2} \cup \left(\bigcup_{j=0}^{k-2} B_o\left(v_j, \max\{\omega(e_{k-1}), \omega(e_k)\}\right) \right) \cup B_o\left(v_{k-1}, \omega(e_k)\right) \right) \backslash V_{k-1}$$

$$\Leftrightarrow C_k = \bigcup_{j=0}^{k-1} B_o\left(v_j, \max_{j<t\leq k} \omega(e_t)\right) \backslash V_{k-1}$$

On s'arrête ici, et on retrouve un terme connu : $\max_{j<t\leq k} \omega(e_t)$ qui correspond exactement à la définition de notre d_k^-. Nous avons ainsi démontré complètement notre théorème sur le déplacement 1D et par extension sur un déplacement 3D.

4.8 Réflexions sur les limites de l'algorithme

Rappelons tout de même les limites du problème. Ce que nous avons fait jusque là, c'est calculer une zone de déplacement possible d'un point sans que l'arbre que nous construisons ne soit modifié sous des conditions fortes : les précédents points vus par l'algorithme ne bougent pas, et nous nous préoccupons de ne pas modifier l'arbre dans les étapes précédentes de construction.

La suite du problème est donc d'envisager les conséquences sur les étapes futures. Autant dire que la difficulté du problème est bien supérieure. En fait, ce problème est équivalent à l'étude des conséquences de l'étape k au déplacement d'un point ajouté à l'étape i de l'algorithme.

L'idée est donc de rendre les calculs des espaces des déplacements possibles, indépendants de l'étape d'ajout du sommet dans l'arbre. En effet, nous le voyons tout au long de ce chapitre cette dépendance est très forte. De manière théorique, se détacher de cette contrainte est possible.

Pour chaque sommet, il existe une évolution de la construction de l'arbre unique comme nous avons vu en section 4.3 ; mais rien ne nous garantit que ces arbres soient identiques. Est-ce un problème ?

Intuitivement, pas vraiment, si nous étudions une zone de déplacement possible pour tous les sommets, pour toutes les constructions possibles obtenues à partir de l'algorithme alors elle devraient garantir la stabilité de tous les arbres construits. Nous pouvons poser cela comme postulat, qui nécessiterait une étude bien plus conséquente.

4.9 Étude statistique du déplacement des points

Après cette étude théorique, revenons à du concret avec nos maillages. Nous avons vu que nos calculs de r^+ et r^- sont de bonnes approximations théoriques pour le calcul du déplacement d'un point. Il faut savoir maintenant en pratique si ces valeurs sont exploitables, et quel est le comportement sur un maillage "réel".

4.9.1 Analyse sur un maillage

Nous appliquons notre étude théorique sur des maillages composés d'un millier de points. Nous nous sommes limités à des maillages de petite taille pour des raisons de temps de calcul.

Ce qui va nous intéresser ici est la distribution des fonctions r^+, r^- sur un maillage particulier nommé *Horse* qui a été normalisé en fonction de la taille de sa boite englobante (figure 4.7).

Tout d'abord, nous remarquons que la plupart des sommets ne peuvent être déplacés dans aucune direction. Plus de 20% d'entre eux ont un r_i^- très proche de 0, et pour la mesure r_i^+, 30% des sommets ne peuvent pas être déplacés. Plus la valeur du rayon de déplacement est grande, plus le sommet est dit robuste. Les résultats ne sont pas surprenants, ils révèlent la fragilité de la structure de l'ACPME.

Concernant la distribution du nombre de sommets en fonction du rayon de déplacement r^+, on remarque que le nombre d'occurrences diminue très rapidement quand la valeur du rayon augmente. Notons que le maximum de r^+ est d'environ 0.018 alors que pour r^- le maximum est de $0,040$. Ainsi, il est plus facile de déplacer un sommet v_i en direction de son père $f(v_i)$ que de l'éloigner.

De plus, pour le rayon r_i^-, nous notons un pic intéressant autour des valeurs $0,015$ et $0,02$. Avec cette observation, nous considérons deux types de sommets ceux que nous ne pouvons pas déplacer et ceux qu'on peut rapprocher autant qu'on veut de leur père.

4.9.2 Comparaison entre plusieurs maillages

Pour chaque maillage et pour chaque point le constituant nous calculons ces trois valeurs.

Après les observations faites sur le modèle *Horse*, nous comparons les critères r^+ et r^- sur l'ensemble des objets de notre base de données et nous présentons les distributions

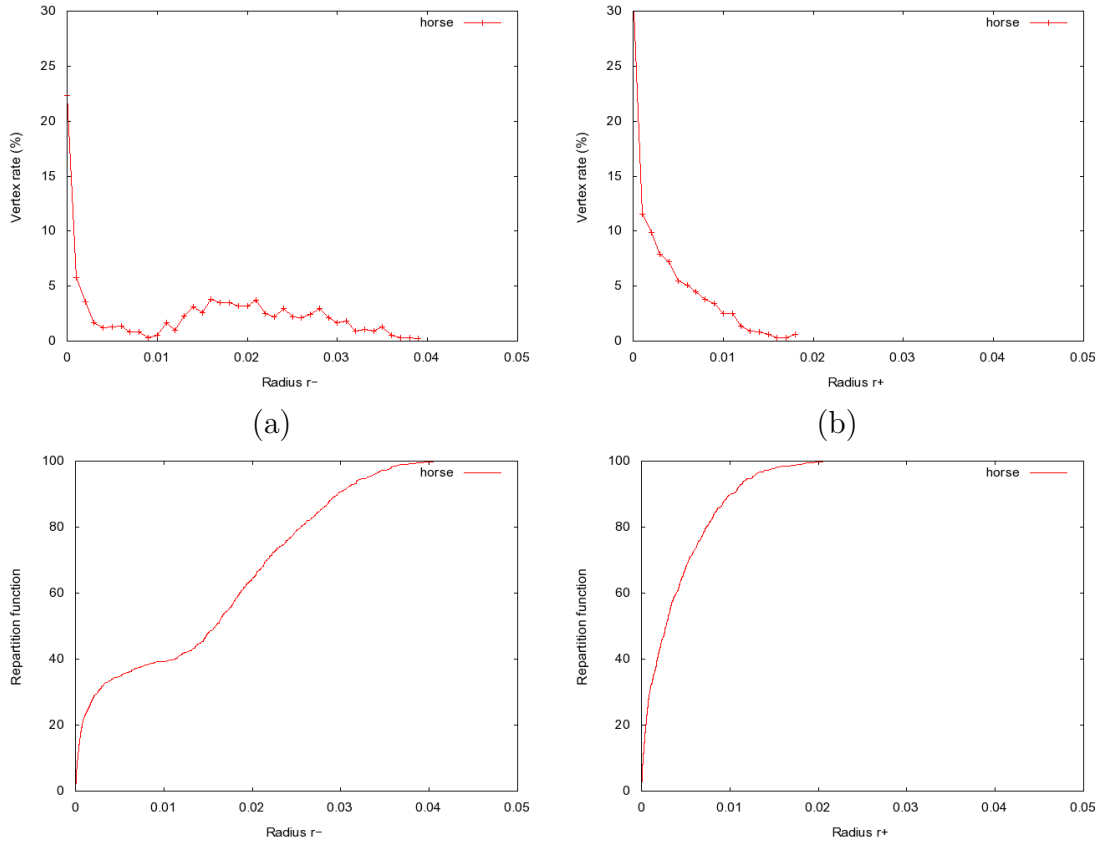


FIGURE 4.7 – Fonction de distribution des rayons de déplacement pour le maillage *Horse* : a) (resp. b)) Distribution de r^- (resp. r^+) pour une normalisation des maillages en fonction de la taille de la boite englobante, c) (resp. d)) Fonction de répartition de r^- (resp. r^+). Les distributions ont été échantillonnées sur un intervalle de 0.001.

des fonctions r^+ et r^- sur la figure 4.8. Les distributions de r^+ et r^- sur des objets qui ont été normalisés en fonction de la taille de la boite englobante (4.8 (a) et (b)), de manière à ce qu'un objet entre dans un cube de taille unitaire. De même, les figures (4.8 (c) et (d)) correspondent aux mêmes distributions mais pour des objets normalisés de façon à ce que la distance moyenne entre les points soit unitaire.

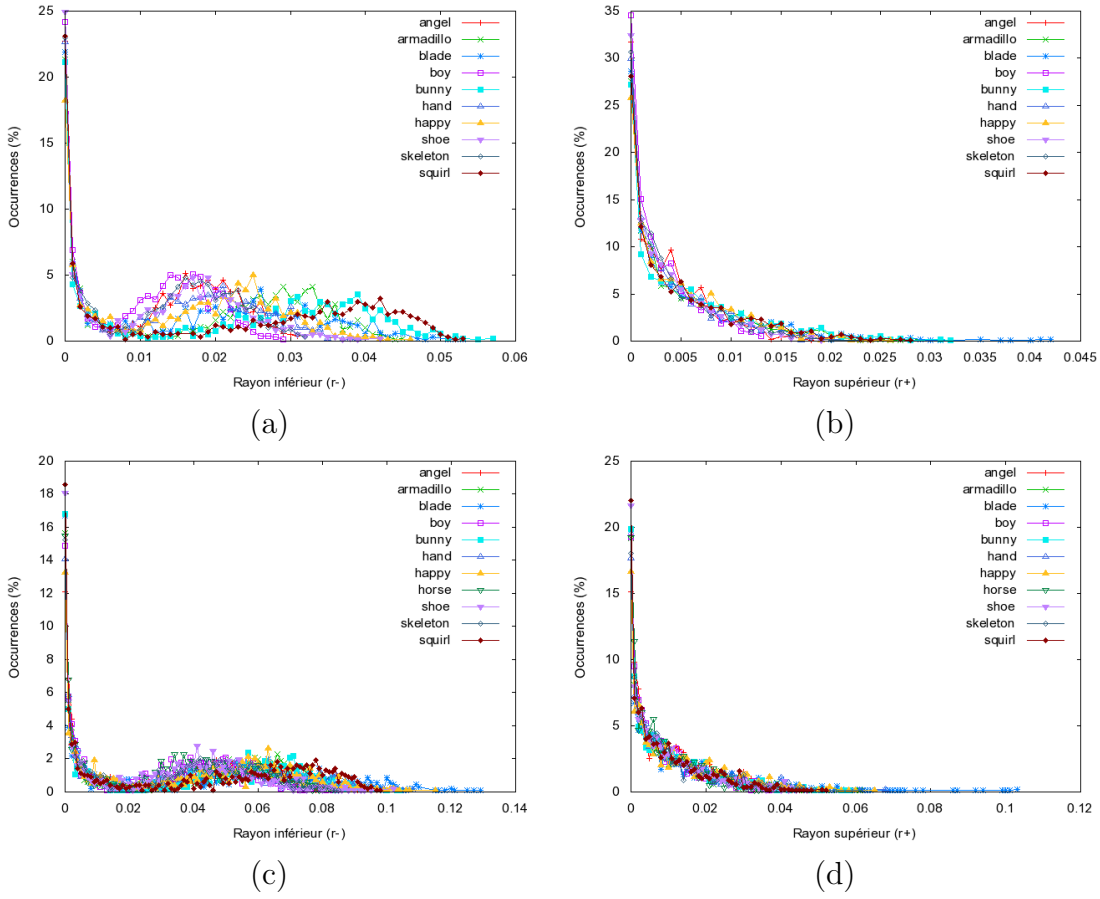


FIGURE 4.8 – Fonction de distribution des rayons de déplacement pour chaque maillage : a) (resp. b)) Distribution de r^- (resp. r^+) pour une normalisation des maillages en fonction de la taille de la boite englobante, c) (resp. d)) Distribution de r^- (resp. r^+) pour une normalisation des maillages en fonction de la longueur moyenne d'une arête. Chaque fonction est échantillonnée sur un intervalle de 0.001.

Le comportement de r^+ est le même pour tous les objets. Un grand nombre de sommets ne peuvent pas se déplacer, et finalement très peu peuvent réellement l'être. Pour r^-, nous remarquons aussi le même comportement que sur l'objet *Horse*. Beaucoup de sommets ne peuvent pas être déplacés, mais nous remarquons toujours la présence de

ce pic autour de la distance moyenne entre deux sommets. En observant la figure (4.8 (c)), nous pouvons valider expérimentalement cette intuition en observant un pic situé au même endroit pour l'ensemble des objets.

Pour savoir si un sommet peut être déplacé, nous proposons de poser un critère r qui quantifie la mobilité de chaque sommet. Ainsi on pose :

$$r_i = \min\{r_i^+, r_i^-\}.$$

La distribution de r est illustrée sur la figure 4.9. La première chose qu'on remarque est que la courbe de r_i est très proche de celle de r_i^+. Ce qui veut dire que la condition d'éloignement est plus contraignante que la condition de rapprochement. C'est d'autant plus dommage que les propriétés de r_i^- étaient intéressantes.

Un autre résultat important qui révèle la grande sensibilité des ACPME, est que 30% des sommets ne peuvent quasiment pas être déplacés. En regardant la courbe et les premières valeurs proches de zéro, on atteint rapidement les 60% de sommets que nous qualifierons de fragiles.

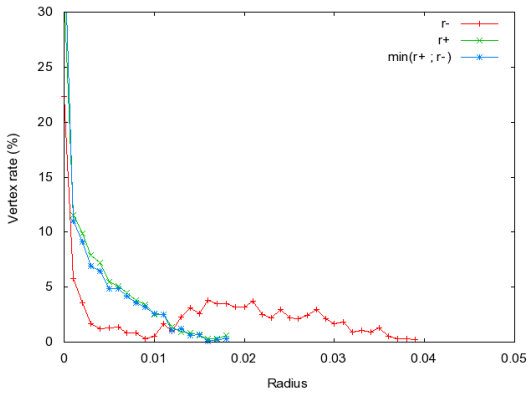


FIGURE 4.9 – Représentation des courbes de r_i, r_i^+ et r_i^- sur un objet (*Horse*).

4.10 Conclusion

Dans ce chapitre nous avons présenté les arbres couvrants de poids minimum euclidien, un outil indispensable dans notre étude. A partir d'un constat de fragilité de la structure à l'ajout de bruit, nous avons analysé cette structure expérimentalement. Le problème global de l'étude de stabilité des ACPME étant $\mathcal{NP}$-complet, nous cadrons le problème en se fixant des hypothèses fortes. Après relaxation des contraintes de l'étude, nous aboutissons à une méthode d'évaluation du déplacement des points validée expérimentalement.

Cette étude nous sert dans le cadre d'application au tatouage 3D, particulièrement à la synchronisation où nous montrons les avantages de cette structure par ses propriétés d'unicité et de facilité de parcours. Les problématiques du tatouage et de la synchronisation sont traitées dans les parties suivantes par un état de l'art et l'intégration des ACPME dans deux nouveaux schémas de synchronisation.

Les résultats présentés dons ce chapitre ont donné lieu à une publication dans une conférence internationale [Tournier *et al.*, 2010] et une publication dans une conférence nationale [Tournier *et al.*, 2009].

Chapitre 5

Synchronisation 3D par les arbres couvrants

Préambule

Précédemment, nous avons calculé l'espace de déplacement possible de chaque point sans modifier les connexions dans l'arbre couvrant de poids minimum calculé (sous certaines conditions). Nous l'avons fait dans le cas où on restreint le déplacement du point à une droite particulière, et sans limite de contrainte. Dans ce chapitre nous présenterons une validation de ce critère pour extraire un sous-ensemble de points dits "robustes", sur lequel nous construirons notre schéma de synchronisation.

Sommaire

5.1 Introduction

Revenons aux critères calculés dans le cas d'un déplacement restreint, nous les prendrons comme référence de calcul. Pour rappel, nous avons d^+ qui est la distance d'éloignement, d^- la distance de rapprochement auxquelles on déduit r^+ et r^- les rayons de déplacement par rapport à la position initiale du point. Et enfin $r = \min\{r^+, r^-\}$ qui est le rayon pour lequel le point d'origine est au centre de l'intervalle.

Dans ce chapitre, nous allons vérifier que r est un critère théorique que nous pouvons exploiter dans la recherche de points "robustes" (5.2). Puis à partir de là nous utilise-

rons les arbres couvrants de poids minimum euclidien (ACPME) dans un schéma de synchronisation que nous détaillerons (5.3). Enfin, nous évaluerons les performances de ce nouveau schéma (5.3.4).

5.2 Corrélation avec le bruitage des points

L'objectif global est d'utiliser les points qui sont le plus mobiles dans un schéma de synchronisation. Bien que nous ayons construit et démontré le déplacement possible d'un point dans un espace précis, les critères calculés ne sont pas pour autant utilisables ainsi.

Pour pouvoir utiliser nos critères dans un schéma de synchronisation, nous avons besoin d'établir une relation entre une fonction issue de notre étude et une mesure expérimentale qui est fonction du taux d'arêtes communes entre l'ACPME d'un nuage de points original et l'ACPME du nuage bruité.

5.2.1 Analyse théorique

Dans un premier temps, nous allons sélectionner les sommets qui sont les plus mobiles, c'est à dire ceux qui ont le rayon de déplacement r maximal. Parmi les sommets les plus mobiles, on en sélectionne $x\%$. Ainsi, on construit $V_{x\%}$ qui est l'ensemble de ces sommets et $r_{x\%}$ la valeur minimale du rayon de déplacement des sommets de $V_{x\%}$.

$$r_{x\%} = \min_{v_i \in V_{x\%}} r_i.$$

Pour constater la fragilité des ACPME, nous avions calculé précédemment le taux d'arêtes communes entre T l'ACPME du maillage original et T_σ l'ACPME du maillage bruité par un bruit gaussien suivant la loi normale $\mathcal{N}(\sigma, 0)$. Nous notons une valeur moyenne $\mu(T, T_\sigma)$ obtenue sur une centaine de maillages bruités.

Ici, nous allons prendre le problème à l'envers et estimer le bruit moyen σ nécessaire pour obtenir un taux d'arête commune égal à $x\%$. Nous noterons cet écart-type $\sigma_{x\%}$, ainsi on a la propriété suivante :

$$\mu(T, T_{\sigma_{x\%}}) = x\%.$$

Nous pensons que s'il y a une corrélation entre les critères de déplacement $r_x\%$ et la puissance de bruit moyen $\sigma_{x\%}$ alors r est un critère de sélection possible dans un schéma de synchronisation.

L'objectif est de trouver une fonction entre $r_{x\%}$ et $\sigma_{x\%}$ tel que $\exists k \in \mathbb{R} : f(\sigma_{x\%}) = k.r_{x\%}$. Nous allons faire cette estimation de manière expérimentale. Pour chaque objet, nous allons calculer, en chaque sommet v_i, la valeur de déplacement possible du point r_i. De là il est facile de déduire $r_{x\%}$, $\forall x$.

Une fois x fixé, on cherche $\sigma_{x\%}$. Pour cela on procède par dichotomie. Une fois σ fixé, on calcule $\mu(T, T_\sigma)$ puis on ajuste le paramètre σ pour se rapprocher de la valeur théorique $\sigma_{x\%}$.

5.2.2 Analyse expérimentale

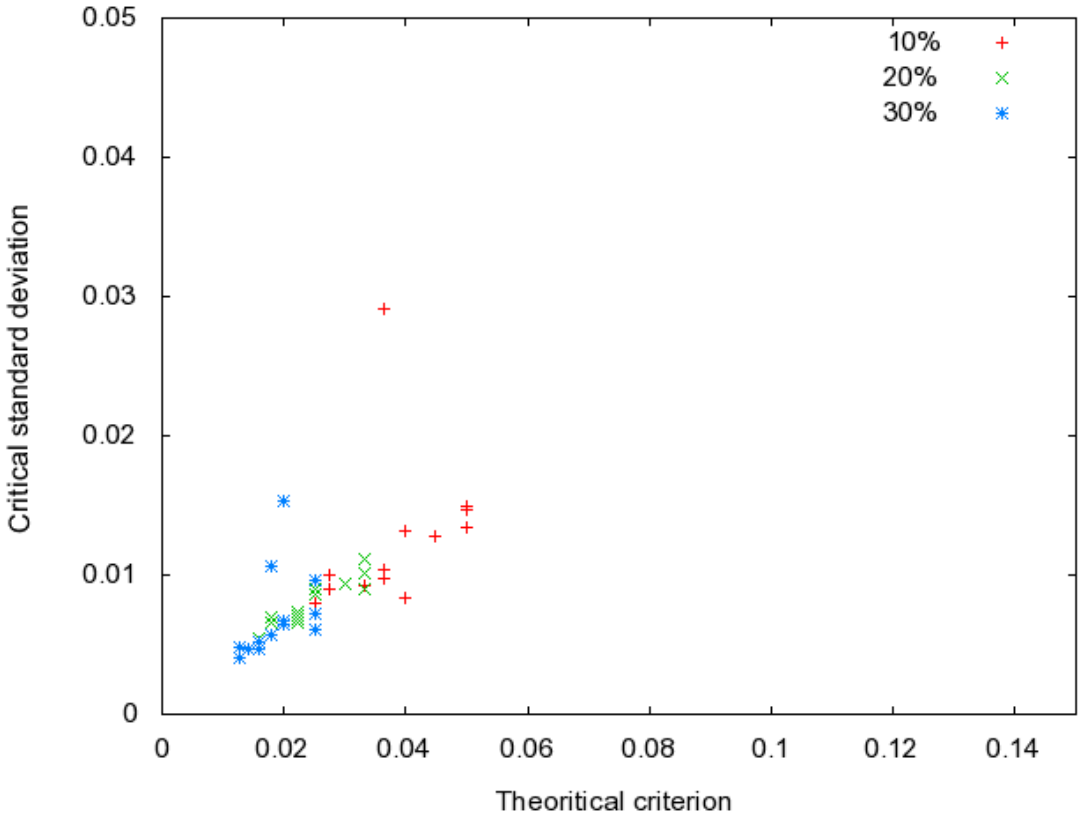


FIGURE 5.1 – Corrélation sur des objets normalisés en fonction de la taille de la boite englobante.

Nous représentons sur les figures 5.1 et 5.2 les résultats de l'expérience. Sur les figures, pour chaque taux de sélection de points robustes x allant de 10% à 30%, chaque point correspond à un objet où nous avons relié $r_{x\%}$ en ordonnée à $\sigma_{x\%}$ en abscisse.

La figure 5.1 correspond aux résultats sur des objets normalisés en fonction de la taille de la boite englobante, tandis que la figure 5.2 correspond à une normalisation en fonction de la longueur moyenne d'une arête du maillage. Ce qu'on peut constater, c'est que nous avons deux groupes de points bien séparés, ce qui est normal puisque r dépend

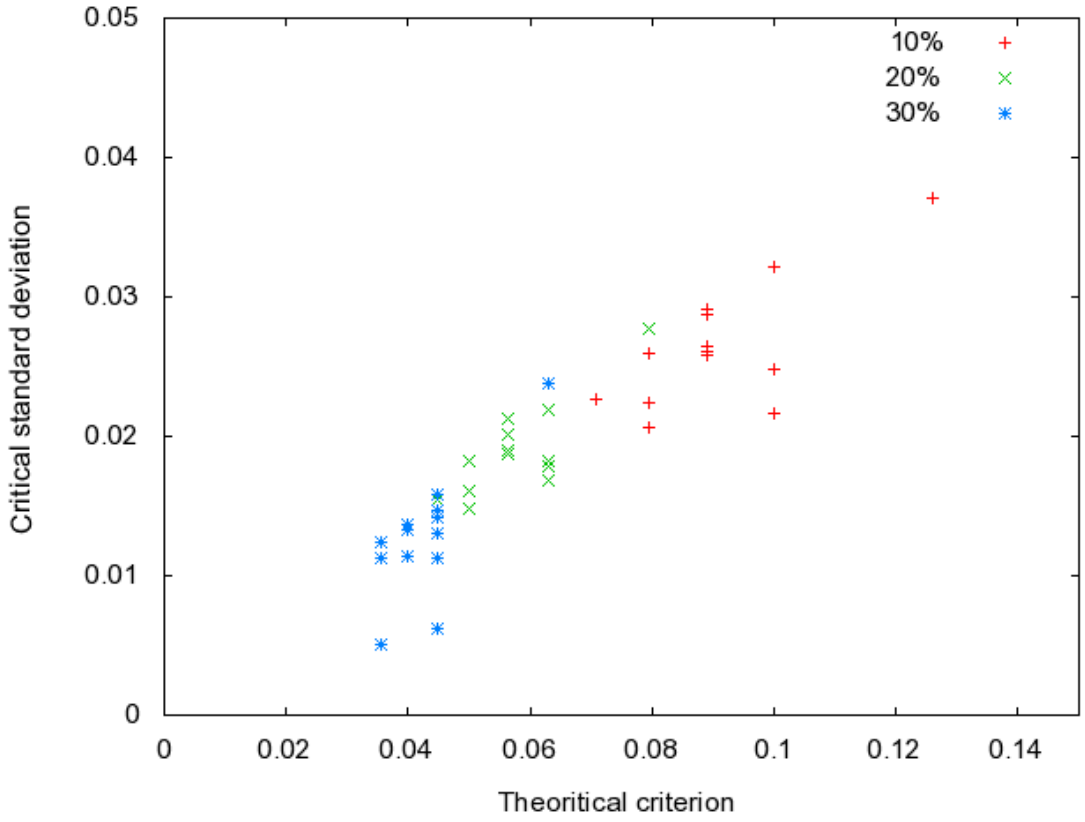


FIGURE 5.2 – Corrélation sur des objets normalisés en fonction de la longueur moyenne d'une arête.

de la taille du maillage ; et dans notre cas lorsqu'on normalise en fonction de la taille de l'objet, il devient généralement plus petit.

De plus, à part une mesure atypique, les points semblent se situer autour d'une droite et que celle-ci ne dépend ni de la normalisation, ni du taux de sélection des points. On s'en rend d'autant plus compte en faisant une régression linéaire sur l'ensemble des points obtenus et en traçant cette droite 5.3.

Cette droite a pour pente une valeur proche de 3, cependant nous n'avons pas démontré théoriquement cette valeur.

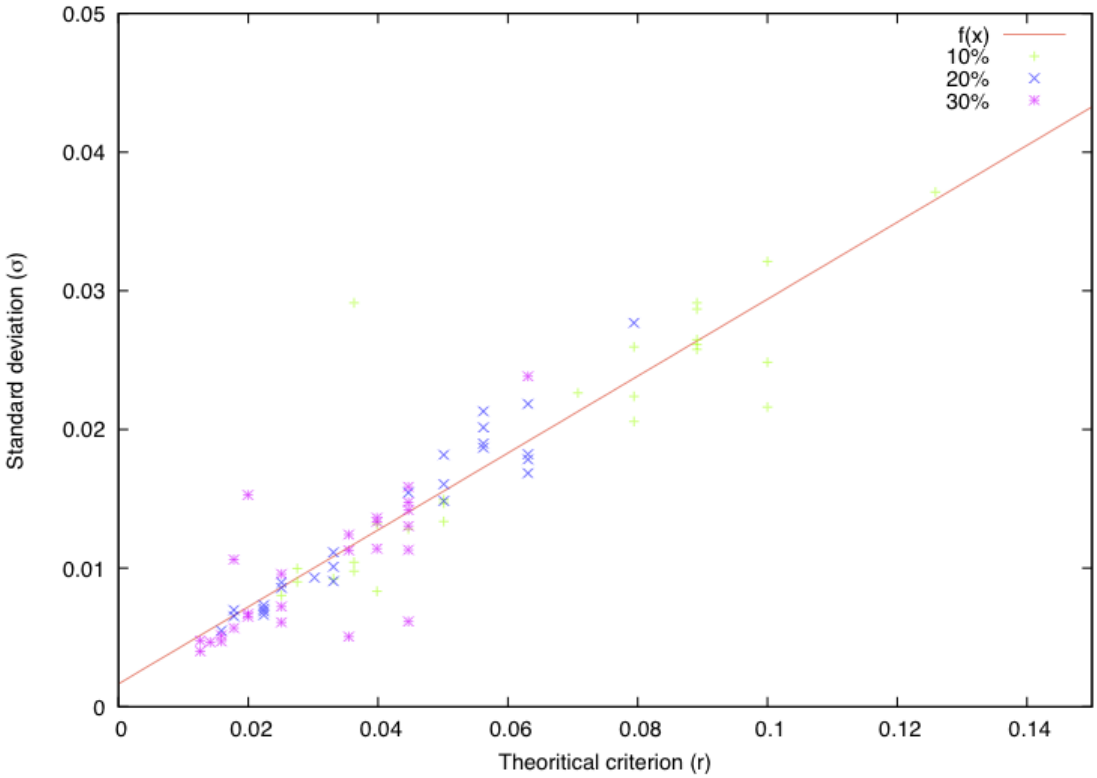


FIGURE 5.3 – Régression linéaire sur l'ensemble des points obtenus expériementalement.

5.2.3 Conclusion

Nous avons bien montré l'existence d'une corrélation entre le rayon du déplacement des points et la puissance d'un bruit gaussien. Ainsi d'un point de vue mathématique, $\exists k \in \mathbb{R}\ k \simeq 3\ f(\sigma_{x\%}) = k.r_{x\%}\ \forall x \leq 30\%$.

Il est important de noter que ce coefficient ne dépend pas de x pour $x \leq 30\%$ et ne dépend pas de la normalisation de l'objet. De plus pour une valeur supérieure à 30%, nous n'obtenons plus de droite. Le comportement des critères devient chaotique. Nous l'expliquons par une sélection de points qui devient elle-même assez difficile à maîtriser, ainsi qu'un rayon $r_{x\%}$ qui devient trop petit et trop proche de la majorité des autres points.

Cette expérience validée, nous allons pouvoir utiliser les ACPME et les sommets les plus mobiles qui correspondent en un sens à des sommets robustes pour le critère de déplacement sans modifier les connexions dans l'ACPME.

5.3 Synchronisation améliorée avec les ACPME

Nous avons constaté la fragilité des ACPME en essayant de comprendre les raisons de cette sensibilité en analysant le déplacement possible des sommets dans l'arbre sans changer les connexions calculées. Maintenant nous allons nous servir de toutes ces expériences pour établir un nouveau schéma qui se veut plus robuste qu'un simple calcul d'ACPME.

Tout d'abord nous allons décrire le schéma que nous comptons mettre en place, puis nous verrons les problématiques auxquelles nous serons confrontés, et pour finir nous exposerons les résultats expérimentaux.

5.3.1 Schéma de synchronisation

Le schéma de synchronisation proposé est entièrement orienté sur des structures de graphes, et donc sur les ACPME que nous avons étudiés tout au long de ce manuscrit. La première étape de l'algorithme consiste à définir la racine de l'arborescence et calculer l'ACPME sur le nuage du points. Puis pour chaque point $v_i \in V$, nous introduisons le calcul de son rayon de déplacement possible $r_i \in \mathbb{R}^+$.

Nous avons vu dans la section précédente qu'il y avait une corrélation entre la puissance de bruit et la sélection des points les plus mobiles. Nous allons donc sélectionner les sommets les plus robustes pour le critère de mobilité dans l'ACPME ; c'est-à-dire les sommets ayant les plus grands r_i. La proportion des sommets sélectionnés est un pourcentage sur le nombre total de sommets qu'on note x (pour ce qui suit, x sera toujours fixé à 20%).

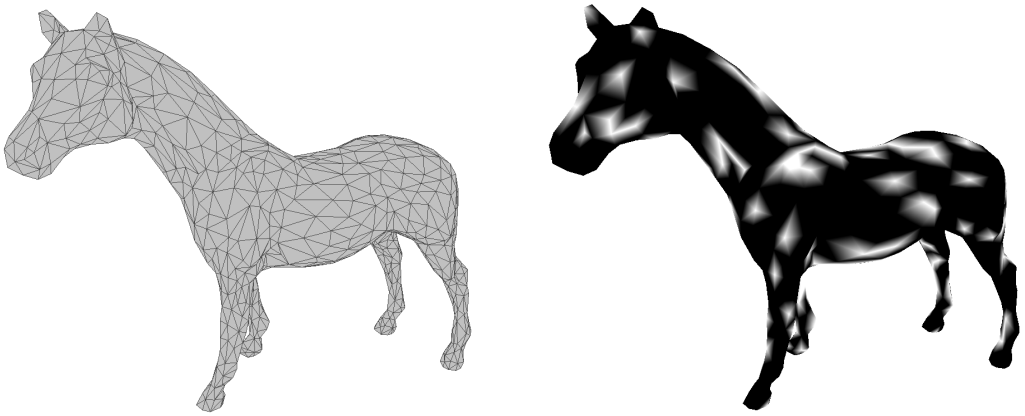


FIGURE 5.4 – Représentation des sommets robustes sur le maillage *Horse* (pour $x = 20\%$). Sur la gauche, nous avons une représentation du maillage et sur la droite nous avons coloré en blanc les zones robustes correspondant aux points sélectionnés.

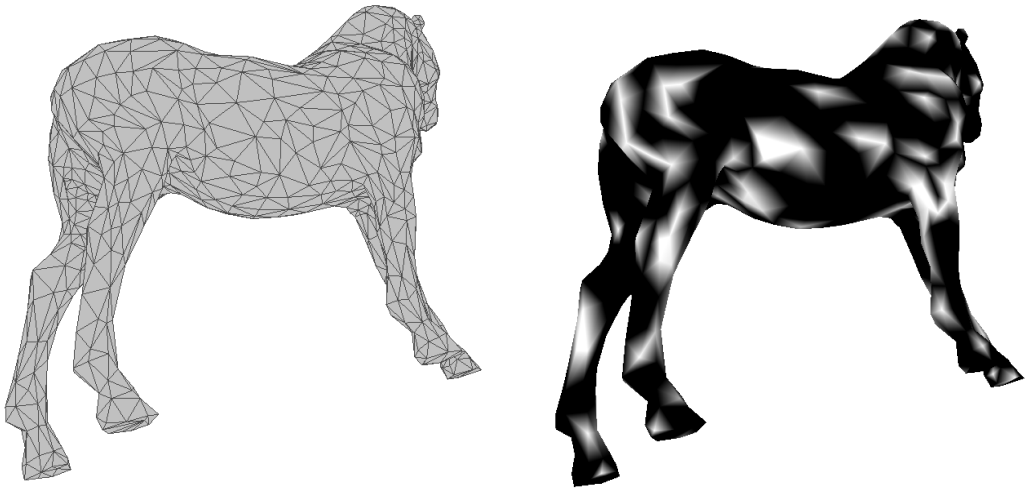


FIGURE 5.5 – Représentation des sommets robustes sur le maillage *Horse* sous un angle différent.

A cette étape nous avons un sous-ensemble de sommets qui sont sélectionnés que nous notons $V_x \subset V$. Nous illustrons sur les figures 5.4 et 5.5, une représentation des points robustes sur le maillage *Horse*.

Comme nous l'avons défini, synchroniser c'est parcourir les données, sélectionner et organiser. Nous avons parcouru l'ensemble des sommets suivant un ordre basé sur la construction de l'ACPME. Nous avons sélectionné un sous-ensemble de points dits robustes. Il ne reste plus qu'à organiser cet ensemble de points.

Organiser, c'est donner un ordre sur ce sous-ensemble de points. Et nous utilisons à nouveau la construction d'un ACPME sur ce sous-ensemble de points. Cet arbre se construit comme précédemment, c'est la distance euclidienne entre les sommets qui est prise en compte. Faisons attention tout de même à un point, la corrélation que nous avons obtenue (section 5.2), ne donne aucune garantie suffisante sur la qualité de la sélection des points. Il se peut que les points sélectionnés ne soient plus les mêmes après une modification du nuage de point.

Pour résumer, notre nouveau schéma de synchronisation, que nous illustrons sur la figure 5.6, repose sur la construction d'un ACPME sur un sous-ensemble de sommets sélectionnés par des critères de mobilités dans l'ACPME du nuage de points. Maintenant nous devons analyser la robustesse de cette méthode, et si elle améliore vraiment la robustesse d'une méthode basée uniquement sur un ACPME calculé sur le nuage de points.

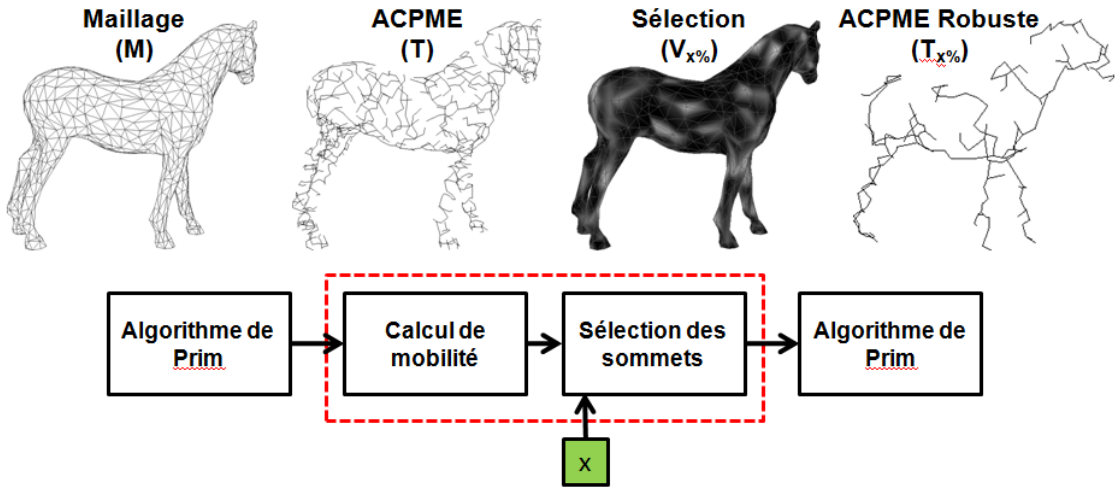


FIGURE 5.6 – Représentation du nouveau schéma de synchronisation basé sur les ACPME.

5.3.2 Protocole d'évaluation du schéma de synchronisation

Pour l'analyse de la robustesse de notre nouvel arbre, nous allons procéder de la façon suivante : tout d'abord, nous prenons notre maillage parmi une base de données que nous avons construites à partir des bases du LGMA, LIRIS, Stanford et l'entreprise STRATEGIES. Les maillages ont ensuite été simplifiés de manière à réduire le nombre de sommets à un millier. Nous avons représenté un échantillon de cette base à la figure 5.7.

Nous faisons l'ensemble des opérations décrites : sélection des sommets robustes (cet ensemble est noté V_x), et construction de l'ACPME sur cet ensemble de point que nous notons T_x. Puis, nous bruitons le maillage original par un bruit Gaussien additif $\mathcal{N}(\sigma, 0)$, on réitère les opérations de sélection des sommets (on notera cet ensemble $V_{\sigma,x}$) et de construction de l'ACPME que nous notons $T_{\sigma,x}$. Nous effectuons ces comparaisons sur une vingtaine de maillages bruités, puis nous faisons une moyenne du nombre d'arêtes communes entre T_x et $T_{\sigma,x}$. Le nombre de comparaisons reste toujours faible en raison de la complexité de l'algorithme.

5.3.3 Analyse du schéma de synchronisation

La question qui se pose est : comment analyser la robustesse de ce nouvel arbre ? Nous avons vu que pour comparer deux arbres construits sur le même ensemble de points, le problème était dit difficile (NP-Complet) ; comment allons-nous comparer T_x et $T_{\sigma,x}$?

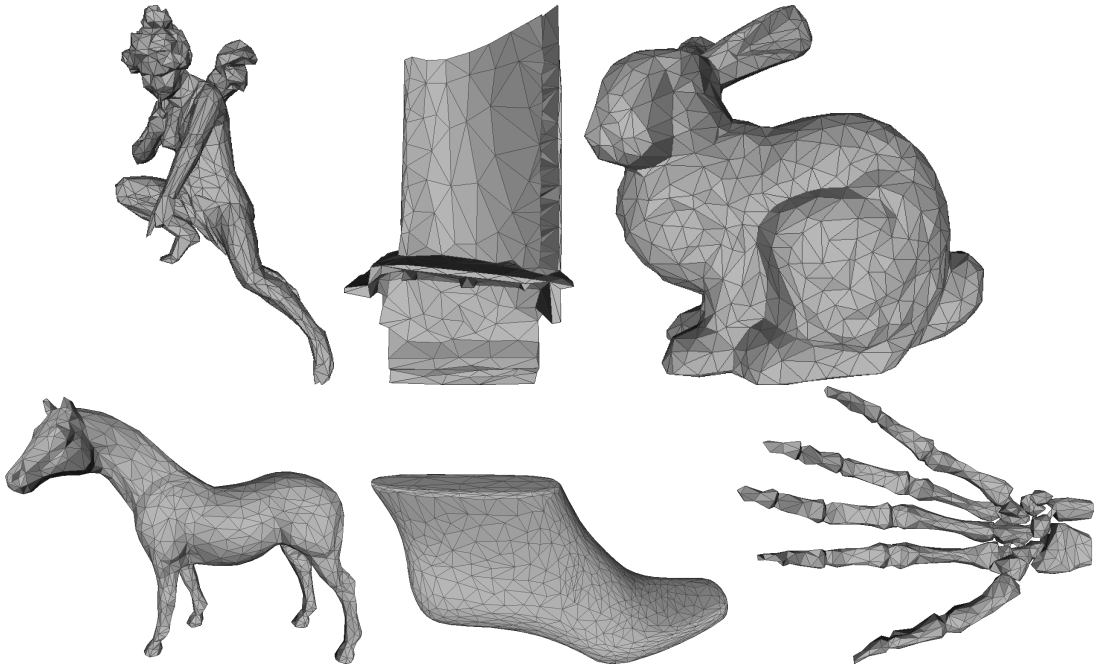


FIGURE 5.7 – Représentation des types d'objets contenus dans notre base personnelle.

Répertorions les deux points sensibles du schéma. Premièrement, et nous l'avons déjà souligné, l'ensemble des points sélectionnés peut-être différent, nous illustrons un exemple de ce cas sur la figure 5.8. Que faire lorsque $V_x \neq V_{x,\sigma}$? Tout va dépendre du schéma d'insertion des données. Nous nous placerons dans le cas le moins favorable, où une différence à la sélection des sommets est une erreur de synchronisation.

Deuxièmement, l'ensemble des sommets est identique, mais l'arbre est différent (figure 5.9). Ici, c'est le cas le plus simple, nous procédons de manière classique à la comparaison des connexions, comme nous l'avions fait lors de l'analyse de la sensibilité des ACPME.

Notons tout de même un troisième cas, où nous avons $V_x \neq V_{x,\sigma}$ mais il existe une bijection entre T_x et $T_{\sigma,x}$ figure 5.10. En d'autres mots, l'arbre est structurellement le même, mais certains sommets sélectionnés ne sont pas identiques. D'un point de vue structurel, la synchronisation est correcte ; d'un point de vue géométrique, faut-il écarter cette solution ?

Nous avons choisi de l'écarter, mais rien n'empêche d'accepter cette solution. Comme nous l'avons dit précédemment, cela dépendra du schéma d'insertion qui vient après la synchronisation.

Concrètement, le cas qui nous posera le plus de problèmes est lorsqu'on aura $V_x \neq V_{x,\sigma}$. Lors de l'expérience, il suffira de vérifier pour chaque connexion que les sommets soient bien les mêmes en ajoutant un identifiant.

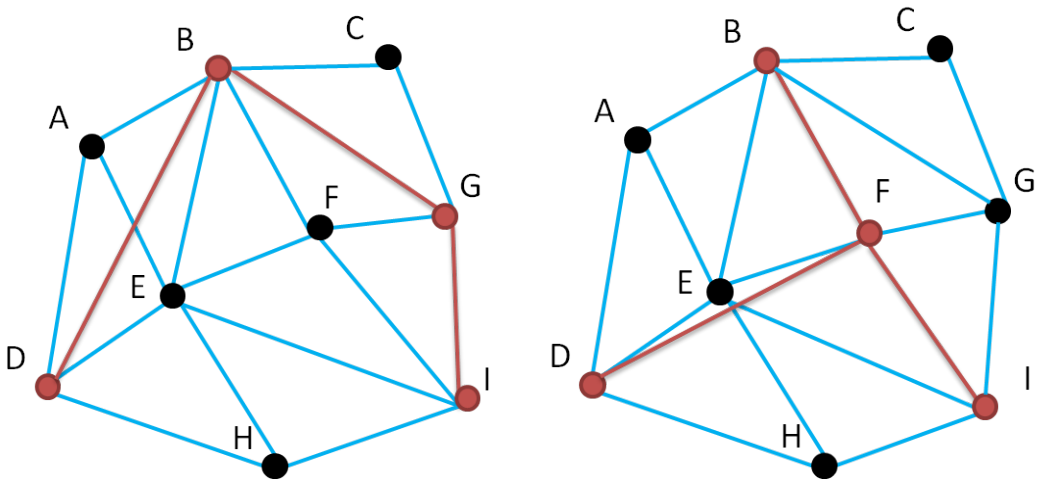


FIGURE 5.8 – Sur la gauche, nous avons un exemple d'un maillage sur lequel nous avons calculé les points les plus mobiles illustrés par des cercles rouge et l'arbre robuste par des segments rouges. Sur la droite, nous avons fait un léger déplacement du sommet F, ce qui a eu pour conséquence de modifier l'ensemble des sommets les plus mobile en rouge.

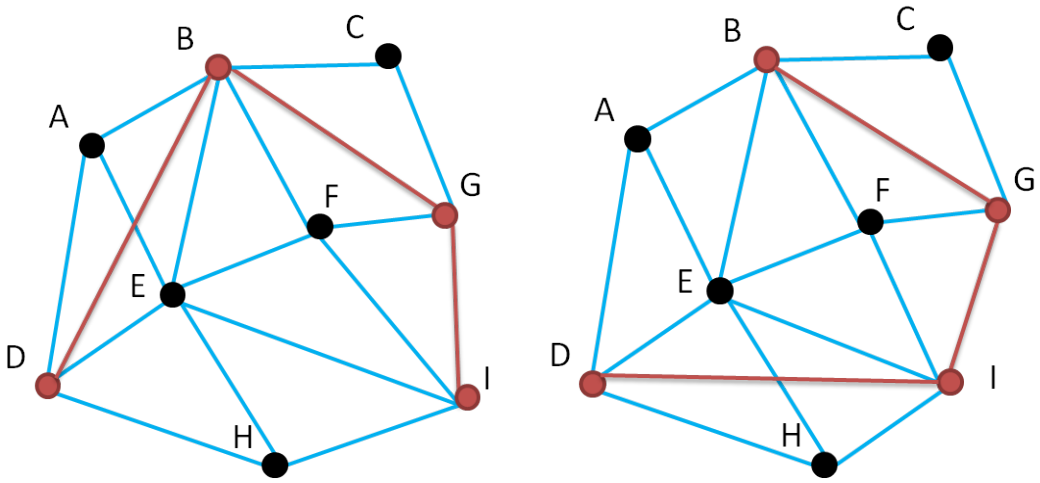


FIGURE 5.9 – Sur la gauche, nous conservons le même maillage. Sur la droite, nous avons déplacé le sommet I, ce qui a eu pour conséquence de modifier les connexions dans l'arbre robuste, mais pas les points sélectionnés.

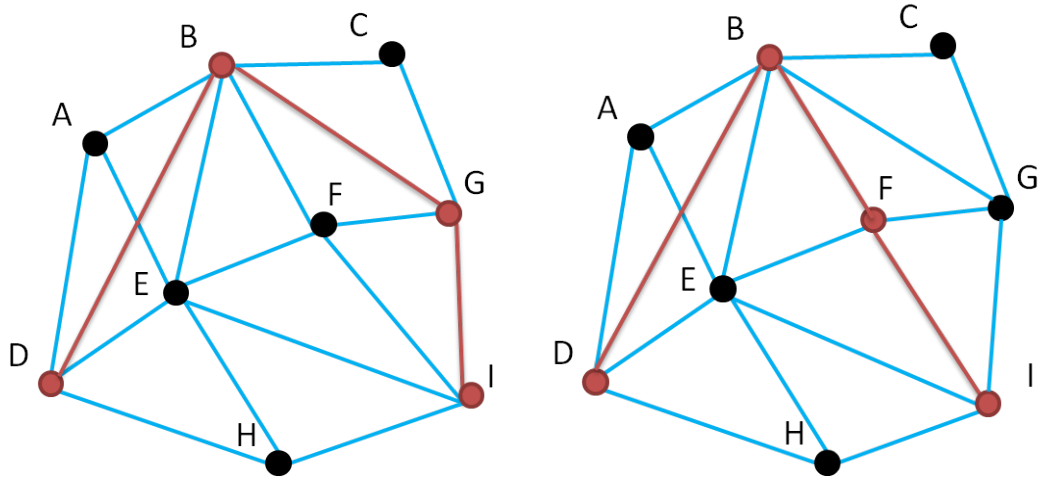


FIGURE 5.10 – Dans ce dernier cas les points sélectionnés différents, mais la structure de l'arbre est identique.

5.3.4 Résultats expérimentaux

La moyenne du taux d'arêtes communes en fonction du logarithme de l'écart type du bruit gaussien ($\log(\sigma)$) est illustré sur la figure 5.11. Dans ce cas nous n'avons pas normalisé les objets, par contre nous avons modulé la puissance du bruit par la distance moyenne entre deux points.

Les résultats obtenus sont mitigés. En effet, pour un bruit additif Gaussien d'écart-type $\sigma = 10^{-5}$, la plupart des maillages ont une correspondance supérieure à 90%. Cependant pour une décade supplémentaire, nous avons une chute vertigineuse en dessous de 40%. On retrouve ce phénomène dans l'analyse de la sensibilité de l'ACPME, pour des bruits de puissance plus faible. Au final, en comparaison avec un ACPME, l'arbre que nous avons construit de manière à ce qu'il soit plus robuste permet de nous faire gagner une décade sur la puissance du bruit.

En dehors de cette courbe générale, nous avons deux cas extrêmes. Les courbes bleues correspondant aux objets *happy et bunny* qui subissent une dégradation du taux de synchronisation une décade avant et la courbe verte de l'objet *blade* qui résiste étonnamment. Pour ces cas, nous n'avons pas d'explication que nous avons pu vérifier.

5.3.5 Conclusion

Nous avons présenté dans cette section un nouveau modèle de synchronisation par les arbres couvrants de point minimum euclidien. En nous basant sur l'analyse du déplacement possible des points dans un ACPME, nous avons validé expérimentalement la sélection de sommets robustes en nous basant sur la mobilité des points dans l'ACPME.

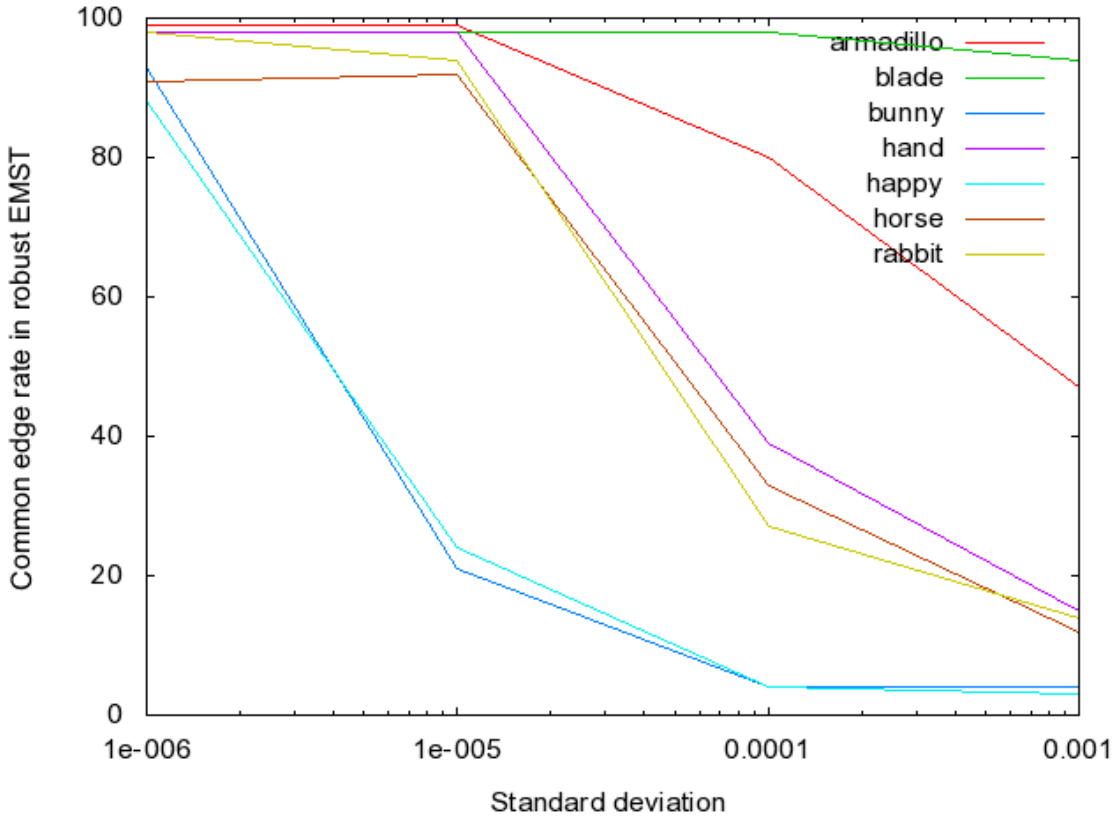


FIGURE 5.11 – Représentation du taux d'arête commune moyen des ACPME robustes ($\mu(T_{x\%}, T_{\sigma,x\%})$) en fonction du logarithme de l'écart type du bruit gaussien ($\log(\sigma)$). L'expérience a été réalisée sur les maillages *armadillo, blade, bunny, hand, happy, horse et rabbit* sur un nombre de 20 itérations.

Malgré les bons résultats, la robustesse du schéma de synchronisation s'avère insuffisant. Nous avons relevé l'ensemble des problèmes qui pourraient surgir (cf. 5.3.3), ce qui pourrait être des pistes pour améliorer le schéma. En tous les cas, nous sommes toujours confronté au problème de complexité algorithmique de calcul des ACPME qui est chronophage sur des maillages supérieurs à 10000 sommets.

Les résultats présentés dans ce chapitre ont donné lieu à une publication dans une conférence internationale [Tournier *et al.*, 2011b].

Dans la partie qui suit, nous conclurons ce manuscrit par des propositions pour améliorer ce que nous avons mis en place.

Troisième partie

Conclusion

Chapitre 6

Conclusion et perspectives

6.1 Bilan sur les travaux de thèse

Dans ces travaux de thèse, nous sommes partis de l'idée introduite par [Amat *et al.*, 2010] d'une synchronisation par les arbres couvrants de poids minimum euclidien. L'objectif a été d'analyser puis d'étudier la structure afin de pouvoir expliquer et cibler ses inconvénients. Nous avons débuté l'étude par le constat de la fragilité des ACPME (cf. chapitre 4). Puis, nous avons développé une étude théorique sur le déplacement possible des points dans le maillage sans modifier les connexions dans l'arbre (cf. sections 4.5 à 4.7). Cette analyse a fait appel aux précédentes études réalisées uniquement dans le cas de graphes quelconques (cf. section 3.4).

Arrivé au bout de cette analyse, nous avons souhaité nous en servir pour améliorer le schéma de synchronisation original. Nous nous sommes orientés vers un schéma de synchronisation sur des zones caractéristiques du maillage. Afin de continuer l'analyse des ACPME, nous avons choisi pour critère la mobilité des points dans le maillage sans modifier les connexions dans l'ACPME. Ce choix est discutable, sachant tous les problèmes de calculs que nous avions. Mais nous avons voulu poursuivre dans cette voie, afin de l'exploiter au maximum et épuiser toutes les possibilités que nous pensions envisageables dans ce domaine. Nous avons eu des résultats meilleurs qu'un schéma classique, mais toujours insuffisant et inutilisable sur des données réelles. Néanmoins ce schéma reste très intéressant. Il peut être réutilisé en changeant les critères de sélection de points pour en faire, nous l'espérons, un schéma bien plus efficace.

Utiliser un ACPME sur des données réelles n'est pas faisable. Cependant l'ACPME est un outil de synchronisation très intéressant. Il offre un parcours original des sommets, et sa structure peut être unique pour un ensemble de points défini. Sa faiblesse étant le calcul sur un nombre de sommets importants, il est tout à fait exécutable sur moins de 1000 sommets pour avoir des temps d'attente raisonnables.

6.2 Perspectives proposées

Nous avons commencé à l'évoquer dans le manuscrit, mais ici nous allons tenter de voir les options possibles à l'amélioration des propositions que nous avons faites. Tout d'abord avec l'algorithme lui-même du calcul des ACPME (6.2.1). Puis une proposition en utilisant notre schéma de construction d'arbre robuste en modifiant les critères de sélection des sommets (6.2.2). Cette proposition, bien que grandement perfectible, a donné lieu a une publication dans une conférence internationale [Tournier *et al.*, 2012].

6.2.1 Complexité et ACPME

Le problème du calcul des ACPME est un problème polynomial, de complexité $o(n^2)$, avec n le nombre de sommets. D'un point de vue théorique, le problème est facile, rapide en temps par rapport à la taille de la donnée et donne un résultat exact. Pour un maillage d'un millier de sommets, le temps de calcul est de l'ordre de la minute. Ce temps peut être acceptable, si l'application garantit un résultat fiable. Dans le cas réel, nous sommes plutôt proche du millions de sommets ; autant dire qu'il devient impossible de pouvoir exécuter correctement cet algorithme. Pourtant, plusieurs solutions sont possibles, et nous en avons testées quelques unes.

ACPME locaux, LMST et généralisation des LMST

Tout d'abord, nous pourrions quadriller l'espace et calculer des sous-arbres à l'intérieur de chacun de ces espaces puis relier chacun de ses arbres les uns aux autres en tentant de conserver les bonnes propriétés d'un arbre. Nous avons testé cette méthode sur des maillages de l'ordre de 10000 sommets. Dans ce cas la résolution du problème se fait dans l'ordre de la seconde. Résoudre un problème avec une donnée dix fois plus grande en cent fois moins de temps, il y a de quoi se poser des questions.

En fait la structure obtenue n'est pas un ACPME. Suivant comment les liaisons sont faites entre les espaces, il se peut que la structure ne soit même pas un arbre. Cependant nous avons une structure qui vérifie dans ces espaces les propriétés d'un ACPME.

Cette construction ressemble a une généralisation des LMST (Local Minimum Spanning Tree), construite de manière empirique. Pour rappel, pour chaque point on calcule un ACPM sur son voisinage ; et que l'on fait l'union de tous ces arbres, on obtient le LMST. Si on veut généraliser le LMST de manière plus rigoureuse, il faudrait prendre le k-voisinage de chaque sommet.

Structures sur les maillages

Nous refusions de dépendre du maillage, pourtant calculer un ACPME en ne prenant en compte que les arêtes du maillage diminue considérablement la complexité. En effet,

bien que la complexité soit généralement calculée en fonction du nombre de sommets elle dépend également du nombre d'arêtes m. Elle dépend également de la manière de programmer l'algorithme. Dans notre cas, le voisinage de chaque sommet est une liste d'adjacence. Ainsi les algorithmes de Prim et de Kruskal ont une complexité de $o(m \log n)$.

Quand on prend en compte le nuage de points $m = (n-1)^2 = o(n^2)$, ainsi la complexité de l'algorithme est de $o(n^2 \log n)$. Si on tient compte des arêtes du maillage uniquement $m = o(n)$, et donc la complexité est de $o(n \log n)$. C'est une différence considérable. Mais elle a un prix, en dépendant du maillage, on devient très sensible aux attaques par remaillage, ce que nous avons refusé dès le début. On peut effectivement se demander, s'il ne faudrait pas revenir sur ce rejet.

D'autant plus que d'autres structures, quant à elle très stables, peuvent se construire en se reposant sur le maillage. Nous avons testé la structure RNG (*Relative Neighborhood Graph*), sur laquelle nous avions un taux de synchronisation toujours supérieur à 90% pour des déplacements bien plus important que pour les ACPME. De plus la construction de ce graphe est extrêmement simple, il suffit de supprimer pour chaque triangle l'arête la plus grande. D'un point de vue complexité, l'algorithme est linéaire $o(n)$; donc extrêmement rapide.

Conclusion

Pour conclure sur ces remarques liées à la complexité, nous voyons déjà que beaucoup de choses sont possibles. Nous nous sommes mis volontairement dans un cas complexe, avec des contraintes relativement fortes. Toutefois, nous avons évoqué la possibilité de faire des approximations ; dans le cas du processus de synchronisation, il serait préférable d'éviter ceci, et de se limiter à des résultats exacts et des algorithmes déterministes.

6.2.2 Zones caractéristiques ordonnées par un ACPME

Précédemment, nous avons fait des propositions algorithmiques pour réduire le temps de calcul. Ici nous allons conserver le schéma de synchronisation par des zones caractéristiques que nous avons mis en place et remplacer les critères de sélection des sommets par des notions géométriques facilement calculables. Nous ajoutons à ce schéma, une couche de multi-résolution qui originellement nous servait à réduire le nombre de points à traiter dans le but de réduire le temps de calcul lors de l'utilisation d'ACPME.

Tout d'abord nous allons décrire ce nouveau schéma, sur lequel nous avons eu le temps de faire de rapides expériences qui mériteraient d'être traitées avec plus de rigueur.

Description de la méthode proposée

Supposons que nous disposons d'un ensemble de maillages $\{M_{HR}, M_1, M_2, ..., M_{LR}\}$. M_{HR} est le maillage original et les autres sont obtenus par une analyse multi-résolution. Cette décomposition multi-résolution peut être obtenue par une décomposition en ondelettes, des simplifications de sommets, ou d'autre techniques utilisées en LOD (Level Of Detail) par exemple.

Pour notre étude rapide, nous avons choisi une méthode de contraction d'arêtes (Quadric Edge Collapse Decimation) [Heckbert et Garland, 1997] implémentée dans MeshLab [Cignoni *et al.*, 2008]. Pour cette décomposition nous créons une relation de parenté entre les maillages de chaque résolution. Soient $v_s^j \in M_j$ et $v_t^{j-1} \in M_{j-1}$, v_s^j et v_t^{j-1} sont parents si et seulement si v_t^{j-1} est dans la cellule de Voronoi de v_s^j calculée avec les sommets de M_j. Autrement dit, le sommet le plus proche de $v_t^{j-1} \in M_{j-1}$ est $v_s^j \in M_j$ dans la résolution inférieure.

Dans la décomposition complète, on choisit un maillage M_j qui sera notre support pour les prochains calculs. On appelle ce maillage, le maillage basse résolution, on le note $M_{LR} = M_j$. Sur le maillage basse résolution, on calcule la courbure Gaussienne k_G pour chaque point. Puis comme dans notre schéma de sélection de sommets, on fixe un taux de sélection x, et on note $V_{j,x}$ l'ensemble des points sélectionnés qui maximisent ce critère.

Une fois les sommets sélectionnés, nous devons les ordonner. Pour cela on conserve la construction de l'ACPME en se basant sur $V_{j,x}$ et en utilisant la distance euclidienne. Maintenant à l'aide de la relation de parenté que nous avons construite, pour chaque point sélectionné dans la basse résolution, nous faisons correspondre un ensemble de points de la haute résolution. Ainsi à chaque point sélectionné correspond un *patch*. Ces patchs sont ordonnés de la même façon que les sommets de la basse résolution le sont par l'ACPME.

Résultats expérimentaux

Encore une fois, beaucoup de paramètres entrent en jeu dans ce schéma. Le taux de sélection des sommets $x\%$: plus x est petit, plus les sommets sont robustes, moins ils sont nombreux. Pour être robuste à la découpe par exemple, il n'est pas nécessaire d'avoir beaucoup de sommets sélectionnés, l'essentiel est qu'ils soient répartis dans le maillage de la manière la plus homogène possible. Face à l'ajout de bruit ou des attaques de lissage, la robustesse est plus importante que la quantité ou la répartition des sommets. Nous analyserons les résultats pour des taux de 5% et 10%. Ces taux ont été choisis arbitrairement.

Nous avons en plus le niveau de la décomposition qui sera choisi pour servir de basse résolution. C'est un paramètre qui serait intéressant d'étudier plus rigoureusement car nous devons ici trouver un compromis entre la pertinence du calcul de la courbure et la

stabilité de la basse résolution. Dans notre modèle expérimental, nous ne descendrons que d'un niveau dans la décomposition.

Pour analyser la robustesse de notre schéma nous procédons de la façon suivante :

— On note $M_{HR,\sigma}$, la maillage original bruité par un bruit gaussien additif suivant la loi normale $N(\sigma, 0)$;

— La basse résolution choisie est la même que pour le maillage original ;

— On construit l'ensemble des maillages $\{M_{HR,\sigma}, ..., M_{LR,\sigma}\}$;

— Dans la basse résolution, on calcule les courbures de chaque point et on sélectionne les $x\%$ des sommets qui maximisent le critère ;

— On construit les patchs dans la haute résolution à l'aide de la fonction de parenté.

Pour ce test expérimental, nous prenons l'objet *Angel*, qui est un maillage de l'ordre de 10000 points que nous avons normalisé de manière à ce que sa boite englobante soit unitaire. Nous allons calculer trois critères pour évaluer notre schéma :

— le taux de sommets communs dans la basse résolution et dans les patchs de la haute résolution ;

— le taux de faux positifs (correspondant au taux de sommets qui sont sélectionnés dans le maillage bruité alors qu'ils ne devraient pas l'être) ;

— le taux de faux négatifs (correspondant au taux de sommets qui ne sont pas sélectionnés dans le maillage bruité alors qu'ils le devraient).

Les résultats sont illustrés sur les figures 6.1, 6.2 et 6.3.

En s'intéressant plus au taux de sommets communs, pour un bruit de l'ordre de la précision des données ($\sigma = 10^{-7}$), la synchronisation est quasi parfaite. Le taux de synchronisation est égal à $99,7\%$ pour des sélections à 10% et à 5%.

Pour un bruitage ($\sigma = 10^{-5}$), nous conservons une synchronisation proche de 80% pour une taux de sélection à 10%. Ce qu'on attendait pas trop, c'est que lorsque la sélection est plus restrictive (5%), les résultats sont moins bons.

Conclusion

Pour améliorer cette méthode, nous devons passer plus de temps. Ces premiers résultats rapides sont intéressants. Nous devons changer le niveau de résolution choisi pour la basse résolution. En prenant une résolution encore plus faible, on devrait gagner en performance pour la synchronisation. Le problème sera ensuite une question de temps de calcul si nous souhaitons trop descendre dans les résolutions.

Cette proposition reste originale, c'est une méthode de synchronisation 3D basée sur la sélection de zones caractéristiques couplé à une analyse multi-résolution. C'est l'une des premières méthodes de ce style qui combine ces deux approches avec [Alface *et al.*, 2007].

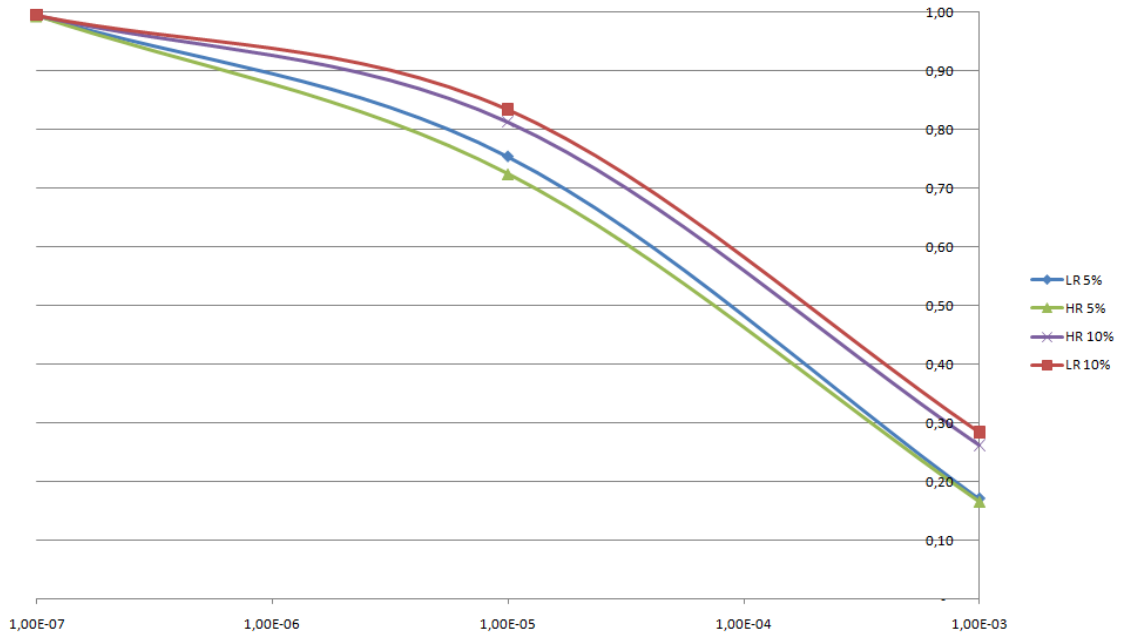


FIGURE 6.1 – . Représentation graphique du taux de sommets communs dans la basse résolution et dans les patchs de la haute résolution en fonction de $\log(\sigma)$.

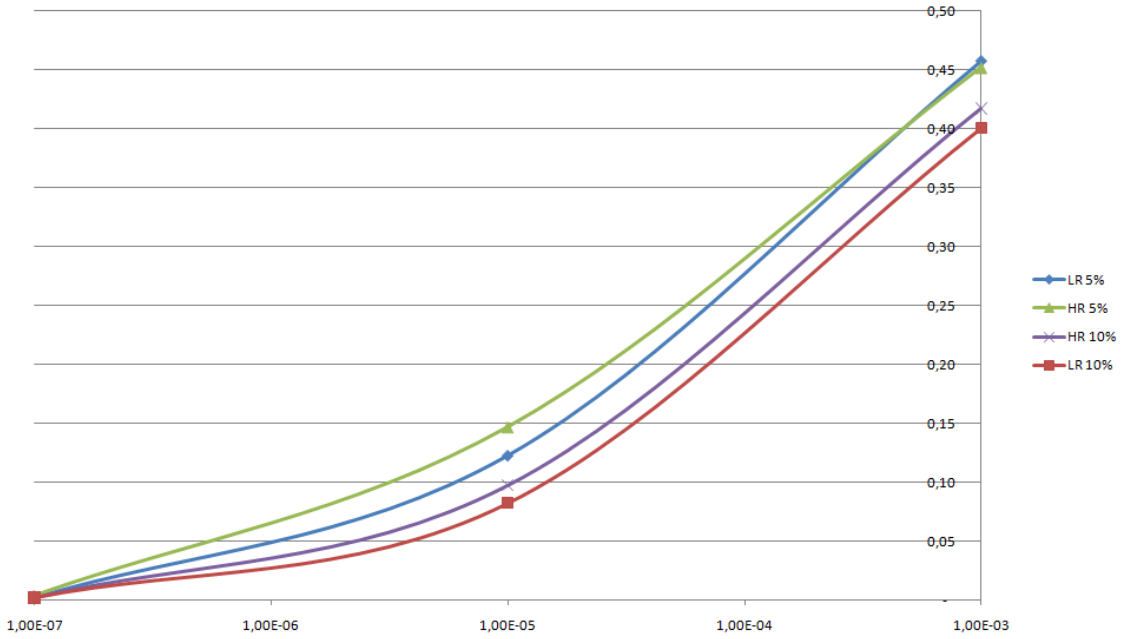


FIGURE 6.2 – . Représentation graphique du taux de faux positifs en fonction de $\log(\sigma)$.

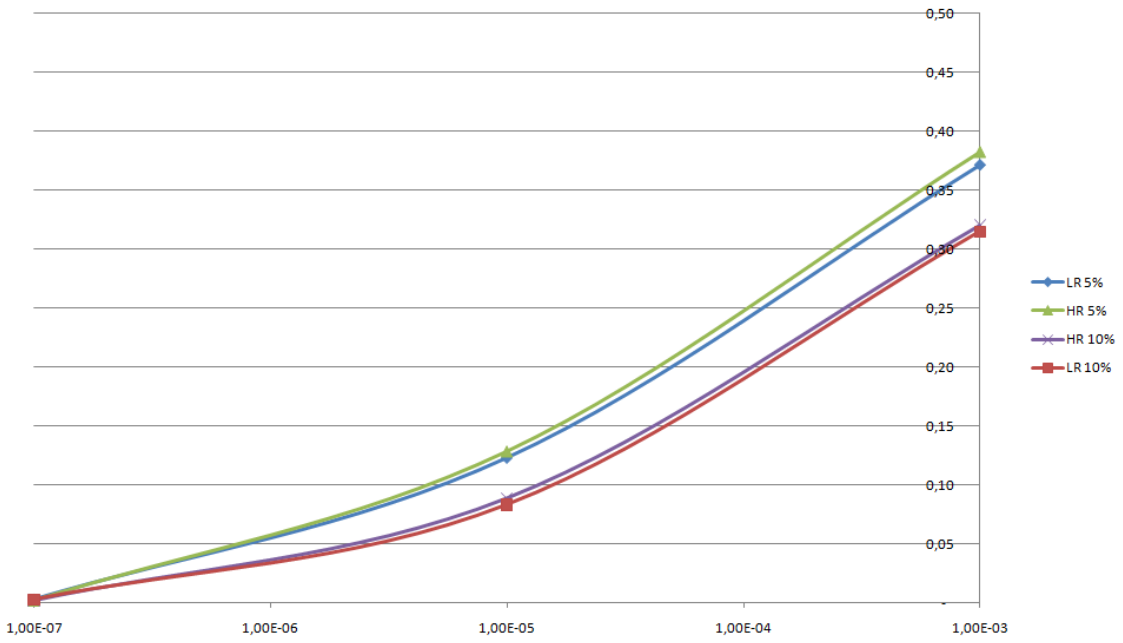


FIGURE 6.3 – . Représentation graphique du taux de faux négatifs en fonction de $\log(\sigma)$.

Bibliographie

P. Agarwal et B. Prabhakaran : Robust blind watermarking mechanism for point sampled geometry. *In Proceedings of the 9th workshop on Multimedia & Security*, pages 175–186. ACM, 2007. Cité page 42.

P.K. Agarwal et J. Matoušek : Relative neighborhood graphs in three dimensions. *In Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 58–65. Society for Industrial and Applied Mathematics, 1992. Cité page 55.

P.R. Alface, B. Macq et F. Cayre : Blind and robust watermarking of 3d models : How to withstand the cropping attack ? *In Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V–465. IEEE, 2007. Cité page 109.

P. Amat, W. Puech, S. Druon et J-P. Pedeboy : Lossless 3d steganography based on mst and connectivity modification. *Signal Processing : Image Communication*, 25 (6):400–412, 2010. Cité pages 46 et 105.

J. S. Armstrong et F. Collopy : Error measures for generalizing about forecasting methods : Empirical comparisons. *International journal of forecasting*, 8(1):69–80, 1992. Cité page 31.

E. Belogay, C. Cabrelli, U. Molter et R. Shonkwiler : Calculating the hausdorff distance between curves. *Information Processing Letters*, 64(1):17–22, 1997. Cité pages 31 et 32.

O. Benedens : Geometry-based watermarking of 3d models. *Computer Graphics and Applications, IEEE*, 19(1):46–55, 1999. Cité page 43.

R. Béniere, G. Subsol, W. Puech, G. Gesquière et F. Le Breton : Decomposition of a 3d triangular mesh into quadrangulated patches. *In International Conference on Computer Graphics Theory and Application (GRAPP 2010)*, pages 96–103, 2010. Cité page 44.

A. Bogomjakov, C. Gotsman et M. Isenburg : Distortion-free steganography for polygonal meshes. *In Computer Graphics Forum*, volume 27, pages 637–642. Citeseer, 2008. Cité page 43.

O. Borůvka : Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí. *Elektrotechnickỳ obzor*, 15:153–154, 1926. Cité page 57.

G. Brassard et P. Bratley : *Fundamentals of Algorithmics*. Prentice-Hall, Inc., 1996. Cité page 42.

P. Cayley : On the colouring of maps. *In Proceedings of the Royal Geographical Society and Monthly Record of Geography*, volume 1, pages 259–261. JSTOR, 1879. Cité page 51.

F. Cayre, P. Rondao-Alface, F. Schmitt, B. Macq et H. Maître : Application of spectral decomposition to compression and watermarking of 3d triangle mesh geometry. *Signal Processing : Image Communication*, 18(4):309–319, 2003. Cité pages 45 et 47.

C.K. Chan et LM Cheng : Hiding data in images by simple lsb substitution. *Pattern Recognition*, 37(3):469–474, 2004. Cité page 26.

P. Cignoni, M. Corsini et G. Ranzuglia : Meshlab : an open-source 3d mesh processing system. *ERCIM News*, 73:45–46, 2008. Cité page 108.

G. Cirio, Lavoué G. et F. Dupont : A Framework for Data-Driven Progressive Mesh Compression. *In* Springer, éditeur : *International Conference on Computer Graphics Theory and Applications (GRAPP), Lecture Notes on Computer Science*, mai 2010. URL http://liris.cnrs.fr/publis/?id=4613. Cité page 23.

M. Costa : Writing on dirty paper. *IEEE Trans. Inf. Theory*, 29(3):439–441, 1983. Cité page 27.

B. Dixon, M. Rauch et R.E. Tarjan : Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21:1184–1992, 1992. Cité pages 67 et 79.

J.J. Eggers, R. Bauml, R. Tzschoppe et B. Girod : Scalar costa scheme for information embedding. *IEEE Transactions on Signal Processing*, 51(4):1003–1019, 2003. Cité page 49.

J. Eisner : State-of-the-art algorithms for minimum spanning trees-a tutorial discussion. 1997. Cité page 56.

L. Euler : Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736. Cité page 51.

K.R. Gabriel et R.R. Sokal : A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259, 1969. Cité page 55.

M. Garland et P.S. Heckbert : Surface simplification using quadric error metrics. *In Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997. Cité page 24.

E.N. Gordeev : Stability analysis in optimization problems on matroids in the metric l1. *Cybernetics and Systems Analysis*, 37:251–259, 2001. Cité page 67.

T. Harte et A. G Bors : Watermarking graphical objects. *In Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 2, pages 709–712. IEEE, 2002. Cité page 45.

T. He, L. Hong, A. Kaufman, A. Varshney et S. Wang : Voxel based object simplification. *In Proceedings of the 6th conference on Visualization'95*, page 296. IEEE Computer Society, 1995. Cité page 24.

P.S. Heckbert et M. Garland : *Survey of Polygonal Surface Simplification Algorithms*. Citeseer, 1997. Cité page 108.

H. Hoppe : Progressive meshes. *In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996. Cité page 24.

Z. Karni et C. Gotsman : Spectral compression of mesh geometry. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286. ACM Press/Addison-Wesley Publishing Co., 2000. Cité page 47.

N. Katoh, T. Tokuyama et K. Iwano : On minimum and maximum spanning trees of linearly moving points. *Discrete and Computational Geometry*, 13(1):161–176, 1995. Cité page 67.

A. Kerckhoffs : La cryptographie militaire. *Journal des sciences militaires*, 9(5-38):161–191, 1883. Cité page 33.

M.S. Kim, S. Valette, H.Y. Jung et R. Prost : Watermarking of 3d irregular meshes based on wavelet multiresolution analysis. *Digital Watermarking*, pages 313–324, 2005. Cité page 49.

F. Kirsch et J. Döllner : Opencsg : A library for image-based csg rendering. *In USENIX Annual Technical Conference, FREENIX Track*, pages 129–140, 2005. Cité page 10.

N. Kiyavash et P. Moulin : On optimal collusion strategies for fingerprinting. *In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE, 2006. Cité page 36.

J.B. Kruskal Jr : On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956. Cité page 57.

W.E. Lorensen et H.E. Cline : Marching cubes : A high resolution 3d surface construction algorithm. *In Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, page 169. ACM, 1987. Cité page 24.

B. Macq et F. Cayre : Data hiding on 3d triangle meshes. *IEEE Transactions on Signal Processing*, 51(4):939–949, 2003. Cité page 43.

X. Mao, M. Shiba et A. Imamiya : Watermarking 3d geometric models through triangle subdivision. *In Proc. SPIE*, volume 4314, pages 253–260, 2001. Cité pages 43 et 46.

C. Monma et S. Suri : Transitions in geometric minimum spanning trees. *Discrete and Computational Geometry*, 8(1):265–293, 1992. Cité page 68.

R. Ohbuchi, H. Masuda et M. Aono : Watermaking three-dimensional polygonal models. *In Proceedings of the Fifth ACM International Conference on Multimedia*, pages 261–272. ACM New York, NY, USA, 1997. Cité pages 40 et 46.

R. Ohbuchi, S. Takahashi, T. Miyazawa et A. Mukaiyama : Watermarking 3d polygonal meshes in the mesh spectral domain. *In Graphics Interface*, pages 9–18. Citeseer, 2001. Cité page 47.

R.C. Prim : Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957. Cité page 57.

Z. Rahmati et A. Zarei : Combinatorial changes of euclidean minimum spanning tree of moving points in the plane. CCCG, 2010. Cité page 68.

S. Rajasekaran : On the euclidean minimum spanning tree problem. *Computing Letters*, 1(1):11–14, 2005. Cité page 57.

J. Rossignac : Edgebreaker : Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics*, 5(1):47–61, 1999. Cité page 43.

J. Rossignac et P. Borrel : Multi-resolution 3d approximations for rendering complex scenes. *Modeling in Computer Graphics*, 455(466):2, 1993. Cité page 24.

M. Salazar-Neumann : The robust minimum spanning tree problem : compact and convex uncertainty. *Operations research letters*, 35(1):17–22, 2007. Cité pages 68 et 79.

M.M. Sales, R. Darazi, J. Giard, P.R. Alface et B. Macq : Feature point-based 3d mesh watermarking that withstands the cropping attack. *In Proceedings of SPIE*, volume 7880, page 78800A, 2011. Cité page 45.

W.J. Schroeder, J.A. Zarge et W.E. Lorensen : Decimation of triangle meshes. *Computer Graphics New York Association For Computing Machinery*, 26:65–65, 1992. Cité page 24.

W. Shuo-zhong et Z. Xin-peng : Recent development of perceptual image hashing. *Journal of Shanghai University (English Edition)*, 11(4):323–331, 2007. Cité page 38.

O. Sorkine : *Laplacian Mesh Processing*. Thèse de doctorat, Tel Aviv University, 2006. Cité page 23.

G. Tardos : Optimal probabilistic fingerprint codes. *J. ACM*, 55:10 :1–10 :24, May 2008. ISSN 0004-5411. URL http://doi.acm.org/10.1145/1346330.1346335. Cité page 38.

R.E. Tarjan : Sensitivity analysis of minimum spanning trees and shortest path trees. *INFO. PROC. LETT.*, 14(1):30–33, 1982. Cité page 67.

G. Taubin : Geometric signal processing on polygonal meshes. *Eurographics State of the Art Reports*, 4(3), 2000. Cité page 23.

N. Tournier, W. Puech, G. Subsol et J-P. Pedeboy : Etude de la robustesse des arbres couvrant de poids minimum dans le cadre du tatouage 3d. Association Francaise d'Informatique Graphique, 2009. Cité page 90.

N. Tournier, W. Puech, G. Subsol et J-P. Pedeboy : Sensitivity analysis of euclidean minimum spanning tree. *In IS&T/SPIE Electronic Imaging*, pages 75260G–75260G. International Society for Optics and Photonics, 2010. Cité page 90.

N. Tournier, W. Puech, G. Subsol et J-P. Pedeboy : 3d data hiding for enhancement and indexation on multimedia medical data. 2nd International Workshop on Medical Image Analysis and Description for Diagnosis Systems, 2011a. Cité page 43.

N. Tournier, W. Puech, G. Subsol et J-P. Pedeboy : Feature vertices for 3d synchronization using euclidean minimum spanning tree. *In IS&T/SPIE Electronic Imaging*, pages 78640W–78640W. International Society for Optics and Photonics, 2011b. Cité page 102.

N. Tournier, W. Puech, G. Subsol et J-P. Pedeboy : 3d multiresolution synchronization scheme based on feature point selection. *In IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2012. Cité page 106.

K. Wang, G. Lavoué, F. Denis et A. Baskurt : Hierarchical watermarking of semiregular meshes based on wavelet transform. *IEEE Transactions on Information Forensics and Security*, 3(4):620–634, décembre 2008. URL http://liris.cnrs.fr/publis/?id=3543. Cité page 48.

K. Wang, G. Lavoué, F. Denis, A. Baskurt et X. He : A benchmark for 3d mesh watermarking. *In 2010 Shape Modeling International Conference*, pages 231–235. IEEE, 2010. Cité page 23.

22es Journées de l'Association Francophone d'Informatique Graphique, Arles, 2009

Etude de la robustesse des arbres couvrant de poids minimum dans le cadre du tatouage 3D

N. Tournier[1,2], W. Puech[1], G. Subsol[1] et J-P. Pedeboy[2]

[1] LIRMM, Univ. Montpellier II, CNRS
161 rue Ada, 34392 Montpellier, FRANCE ;

[2] Strategies S.A., 41-43 rue de Villeneuve, Parc des Affaires SILIC - BP 80429
94583 Rungis, FRANCE.

Abstract

En stéganographie et tatouage 3D, la synchronisation des données à cacher est l'un des principaux problèmes. Nous devons savoir où se trouve le message que nous avons dissimulé dans le médium pour pouvoir l'extraire correctement. De nombreux algorithmes de tatouage 3D ont été proposés ces dernières années. Dans cet article, nous nous intéressons particulièrement à une méthode de synchronisation basée sur le calcul d'arbre couvrant euclidien de poids minimum (EMST : Euclidean minimum spanning tree) sur un nuage de points. Dans notre approche, nous analysons la stabilité des EMST, et nous proposons une méthode théorique de mesure. A partir de cette analyse, nous présentons des résultats sur la localisation et la visualisation de zones robustes utilisables pour une méthode de synchronisation par EMST.

In 3D steganography and watermarking, the synchronization of the hidden data is a major problem. We need to know where the message is embedded in order to extract the correct information. Various algorithms have been proposed for the last couple of years and we focused on a method based on Euclidean minimum spanning tree (EMST) for the mesh vertices. In this paper we analyze the sensitivity of the EMST structure in order to propose a new method more robust. We present and demonstrate the theoretical analysis and we propose our first techniques to predict the robustness of the EMST. Moreover, we can apply this analysis to various applications that can be useful in 3D steganography such fragile area detection and prediction of the 3D object robustness during transmission on a noisy channel.

1. Introduction

De nos jours, Internet est indispensable pour la diffusion de données multimédia. De plus en plus d'objets 3D sont transmis en infographie ou CAO par exemple. Il est important d'une part, de protéger ces données et d'autre part, de pouvoir y associer des informations supplémentaires. L'insertion de données cachées peut répondre à ces attentes.

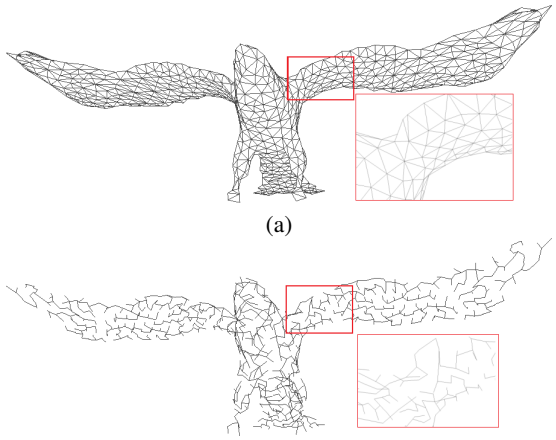
La stéganographie et le tatouage sont deux domaines de dissimulation de contenu, dont une récente étude a été proposée pour la 3D par Wang *et al.* [WLDB08]. L'objectif est de transmettre un message m dissimulé dans un signal hôte c_0 (image, vidéo, objet 3D). Le message m est codé en prenant en compte les caractéristiques du signal hôte c_0. Nous obtenons une marque w_a qui est ajoutée au signal c_0 pour obtenir le signal tatoué c_w qui est transmis sur le canal. A la réception, à partir du signal tatoué transmis c'_w, le message m' est extrait et décodé.

Lors de l'insertion du message nous devons savoir où se sont réparties les informations binaires dans le maillage afin de pouvoir les extraire. Il s'agit du processus de synchronisation des données, un des problèmes majeurs du tatouage et de la stéganographie 3D.

De nombreuses méthodes ont été proposées, et nous nous intéressons à la synchronisation par le calcul d'un EMST proposé par Amat *et al.* [APDP08]. Un EMST $T = (V, E_T)$ est un arbre couvrant de poids minimum (MST : Minimum spanning tree), c'est-à-dire un arbre qui relie tous les sommets d'un graphe $G = (V, E)$ en utilisant les arêtes $e_i =$

$\{v_s, v_t\} \in E$ de poids $\omega(e_i) \in \mathbb{R}^+$ qui minimise le poids total de toutes les arêtes $e_j \in E_T$.

L'originalité de la méthode proposée par Amat *et al.* est de ne déplacer aucun sommet lors du tatouage et de synchroniser le message à l'aide d'un EMST afin de parcourir les sommets du maillage de manière unique. Nous illustrons, Figure 1.a, un maillage et l'EMST correspondant en Figure 1.b.



(a) (b)

Figure 1: *Illustration 3D: a) Maillage 3D, b) EMST correspondant calculé par l'algorithme de Prim [Pri57].*

Afin de dissimuler un message dans un maillage, Amat *et al.* analysent dans l'EMST l'ensemble des quadruples qui vérifient différentes conditions de coplanarité, convexité et chevauchement. Ces quadruples sont utilisés pour le tatouage, et restent détectables après l'insertion du message. Néanmoins, cette méthode est très fragile à toute modification de la position des sommets. C'est pourquoi nous nous intéressons à la sensibilité des EMST dans le but de localiser les zones les plus robustes pour ce genre de synchronisation.

Pour chaque sommet, nous proposons de calculer son déplacement possible tout en gardant les mêmes connexions dans l'arbre final. Le problème étant très difficile, nous ajoutons quelques hypothèses afin de se simplifier les conditions d'étude.

Nous présentons, en section 2, un état de l'art sur la sensibilité des EMST. En section 3, nous proposons notre analyse théorique de la sensibilité des EMST. En section 4, nous montrons des résultats expérimentaux de notre approche. Enfin, nous concluons et présentons des perspectives en section 5.

2. Sensibilité des MST

Selon Gordeev [Gor99, Gor01] la sensibilité des MST est un problème d'optimisation basé sur des représentations mathématiques appelées matroïdes. Soit un graphe $G = (V, E)$ avec n sommets et m arêtes. Gordeev considère le modèle suivant. Soient $D_m = \{\tau_1, \ldots, \tau_q\}$ avec $(q > 1)$, une partie de E appelée trajectoires ; un vecteur $A = (a_1, \ldots, a_m) \subset \mathbb{R}^m$ tel que $\forall i \; a_i = \omega(e_i)$ correspondant aux poids des arêtes du graphe et $\tau(A)$ la longueur de la trajectoire pour un vecteur A avec $\tau(A) = \sum_{e_i \in \tau} a_i$.

Le problème combinatoire est défini par la paire (E, D_m) où A est la variable à optimiser pour minimiser la fonction $\tau(A)$. Avec ce modèle, Gordeev fait l'analogie avec le probleme de calcul d'un MST avec D_m qui est l'ensemble des arbres couvrants de G dans lequel le MST minimise la fonction $\tau(A)$.

Soit $\psi(A)$ l'ensemble des indices i des trajectoires optimales τ_i du problème donné pour un vecteur A, et $B \in \mathbb{R}^m$ tel que $\|B\| < \varepsilon$ un vecteur perturbateur. Gordeev parle de ε-stabilité lorsque $\psi(A+B) \subset \psi(A)$.

En ce qui concerne le problème du calcul d'un MST, pour un bruit d'une intensité donnée ε, il existe quelques arbres qui sont toujours solution du problème après la perturbation. Il en déduit alors un rayon de stabilité $\rho(A) = \sup\varepsilon$ tel que A est ε-stable pour le problème. Ce rayon est calculable avec une complexité de $O(n^3 m \log(\frac{n^2}{m}))$, soit pour un graphe complet $O(n^5)$ $(m = O(n^2))$.

Dans Dixon *et al.* [DRT92], l'analyse de la sensibilité du MST revient à calculer pour toutes les arêtes e_i du maillage, de combien on peut augmenter ou diminuer le poids de l'arête $\omega(e_i)$ sans modifier les connexions dans le MST $T = (V, E_T)$. Ce problème est divisé en deux parties.

Tout d'abord pour les arêtes e_i qui n'appartiennent pas à l'arbre, ils calculent la diminution possible du poids de ces arêtes sans affecter la minimalité de l'arbre T. Puis, pour toutes les arêtes e_i qui sont dans l'arbre, ils augmentent le poids de ces dernières sans modifier les connexions du MST. Ces calculs sont faisables en un temps linéaire en fonction du nombre d'arêtes.

Dans cette section, nous avons montré deux approches analysant la sensibilité des MST. Nous rappelons que nous traitons des données géométriques donc :

- $G = (V, E)$ est un graphe complet, chaque sommet est lié aux autres. Donc $m = n(n-1)$, ce qui maximise la complexité des algorithmes classiques de la théorie des graphes ;
- les poids des arêtes $\omega(e_i)$ ne sont pas indépendants. Si $\omega(e_i)$ est modifié, cela implique qu'au moins un des deux sommets composant l'arête $e_i = \{v_s, v_t\}$ est déplacé. Dès qu'on déplace un sommet du graphe, les poids des $(n-1)$ arêtes le reliant aux autres vont également être modifiés.

Pour Gordeev et Dixon *et al.* les poids des arêtes $\omega(e_i)$ sont indépendants, c'est pourquoi leurs approches ne sont pas applicables dans notre cas. Nous proposons, en section 3, une nouvelle analyse de la sensibilité des EMST.

3. Analyse théorique de la sensibilité des EMST

Soit $G = (V,E)$ un graphe complet valué avec V l'ensemble des sommets et E l'ensemble des arêtes tel que :

- chaque sommet du maillage est un sommet du graphe $v_i \in V$;
- $\forall v_s, v_t \in V \ ; \ v_s \neq v_t \ ; \ \exists e_i = \{v_s, v_t\} \in E$ et $\omega(e_i) = d(v_s, v_t)$, la distance euclidienne entre deux sommets du maillage.

3.1. Calcul des MST : rappels et notations

Pour un graphe valué connexe $G = (V,E)$, nous proposons de calculer son EMST $T = (V, E_T)$. Le calcul des MST est un problème algorithmique bien connu. Pour nos travaux, nous avons choisi d'utiliser l'algorithme de Prim [Pri57] car c'est un algorithme qui permet de construire incrémentalement l'EMST en ajoutant à chaque étape une arête dans l'arbre. Nous avons donc à chaque étape l'EMST d'un sous ensemble de V.

L'algorithme commence à partir d'un sommet initial $v_0 \in V$. A l'étape i de l'algorithme, est ajouté à l'arbre courant $T_i = (V_i, E_i)$ le sommet le plus proche v_i de l'ensemble V_{i-1}. Nous construisons ainsi une suite d'arbres $(T_i)_{0<i<n} = (V_i, E_i)_{0<i<n}$ dépendant du sommet source v_0 tel que $(V_i)_{0<i<n}$ soit une suite d'ensembles de sommets et $(E_i)_{0<i<n}$ une suite d'ensembles d'arêtes :

- $V_0 = \{v_0\}$;
- $E_0 = \emptyset$;
- $V_i = V_{i-1} \cup \{v_i\}$ $(\forall i < n)$, avec $v_i \in \overline{V_{i-1}} = V \backslash V_{i-1}$ le sommet le plus proche de V_{i-1}, défini par l'équation (1) ;
- $E_i = E_{i-1} \cup \{v_i, f(v_i)\}$ $(\forall i < n)$, avec $f : V_i \to V_{i-1}$ une fonction qui donne le sommet le plus proche de v_i dans V_{i-1} (le "père" de v_i), défini par l'équation (2).

$$v_i = \underset{v_j \in \overline{V_{i-1}} \ ; \ v_k \in V_{i-1}}{\text{Argmin}} \omega(v_j, v_k); \tag{1}$$

$$f(v_i) = \begin{cases} \underset{v_j \in V_{i-1} \ ; \ v_k \in \overline{V_{i-1}}}{\text{Argmin}} \omega(v_j, v_k), & i > 0 \\ v_0, & i = 0. \end{cases} \tag{2}$$

Avec ces notations, l'ESMT de G est l'arbre $T = T_{n-1}$ dont nous étudions la sensibilité. Nous voulons savoir pour tous les sommets v_i de T comment les déplacer dans l'espace en modifiant leurs coordonnées $\{x_i, y_i, z_i\}$, tout en gardant les mêmes connexions dans l'arbre.

3.2. Analyse de la sensibilité des EMST et simplifications du problème

Etant donné $G = (V,E)$, nous calculons son EMST et à chaque étape de l'algorithme de Prim nous avons $T_i = (V_i, E_i)$. Nous fixons une étape i, nous déplacons le sommet v_i et notons v^* le nouveau sommet. Soit G^* le graphe dont le sommet v_i a été déplacé en v^*. Ainsi $\forall i$, on note $T_i^* = (V_i^*, E_i^*)$ l'EMST calculé par l'algorithme de Prim sur le graphe G^*.

Comme il est difficile d'analyser la sensibilité d'un EMST en considérant que la totalité des sommets peut être perturbée, nous posons des simplifications sur la perturbation des sommets afin de rendre le problème plus abordable :

- à l'étape i de l'algorithme de Prim, seul le sommet v_i est pertubé en v^* ;
- la perturbation est restreinte à la demi-droite $]f(v_i); v_i)$.

Le long de cette demi-droite $]f(v_i), v_i)$, nous voulons connaitre de combien il est possible de rapprocher et d'éloigner v_i de son "père" $f(v_i)$ sans changer les connexions dans l'arbre à l'étape i.

Avec les notations précédentes, on suppose que T_i^* vérifie l'hypothèse suivante : $\forall k$, $k < i$; $T_k = T_k^*$, c'est-à-dire que les connexions dans l'arbre ne sont pas modifiées jusqu'à l'étape $(i-1)$. Et nous voulons qu'à l'étape i :

- $v^* = v_i^*$, le sommet déplacé à l'étape i est toujours sélectionné à cette étape après le déplacement ;
- $f(v^*) = f(v_i)$, le "père" du sommet sélectionné à l'étape i reste le même.

3.2.1. Distance limite minimum (d^-)

Soient $f(v_i)$, le père de v_i et V_k $(k < i)$ le plus petit ensemble au sens de l'inclusion qui contient $f(v_i)$, $f(v_i) \in V_k$ et $f(v_i) \notin V_{k-1}$ (*cf.* Figure 2). Par construction, à l'aide de l'algorithme de Prim, nous montrons que : pour toutes les arêtes de l'EMST $e_j \in E_{i-1} \backslash E_k$ de poids $\omega(e_j)$ vérifient :

$$\forall e_j = \{v_s, v_t\} \in E_{i-1} \backslash E_k \ ; \ \omega(e_j) < \omega(f(v_i), v_i). \tag{3}$$

A partir de cette remarque, nous pouvons déduire la limite de la distance minimum d_i^- entre v^* et $f(v_i)$ afin de respecter les hypothèses de stabilité de l'EMST :

$$d_i^- = \max_{e_j \in E_{i-1} \backslash E_k} \omega(e_j). \tag{4}$$

Donc pour garder les mêmes connexions dans l'EMST à l'étape i, v^* doit vérifier :

$$\omega(f(v^*), v^*) > d_i^-. \tag{5}$$

L'équation (5) est démontrée en annexe A :

3.2.2. Distance limite maximum (d^+)

Le calcul de la distance limite maximum se divise en deux parties. Tout d'abord, pour sélectionner le sommet v^* à

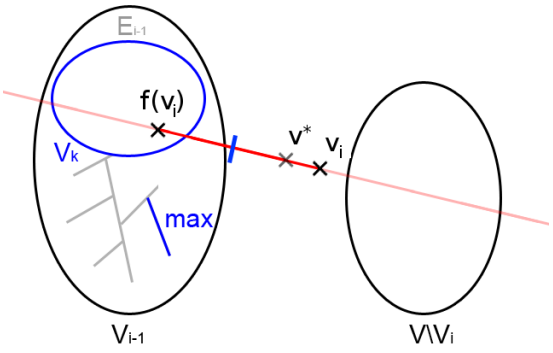


Figure 2: *Schéma du calcul de la distance limite minimum (d^-).*

l'étape i dans l'algorithme de Prim, nous devons chercher le deuxième sommet le plus proche de l'ensemble V_{i-1} que nous noterons $s(v_i)$ (*cf.* équation (6)). Ceci nous permet d'obtenir une première limite candidate. Trivialement, si v_i s'éloigne trop de l'ensemble V_{i-1} (tel qu'il soit plus loin que $s(v_i)$) alors c'est le sommet $s(v_i)$ qui est le plus proche de V_{i-1}. La Figure 3.a illustre ce premier calcul.

$$s : V_i \to \overline{V_i}.$$

$$s(v_i) = \begin{cases} \underset{v_j \in \overline{V_i}\ ;\ v_k \in V_{i-1}}{\text{Argmin}} \ \omega(v_j, v_k), & i < n-1 \\ v_{n-1}, & i = n-1. \end{cases} \tag{6}$$

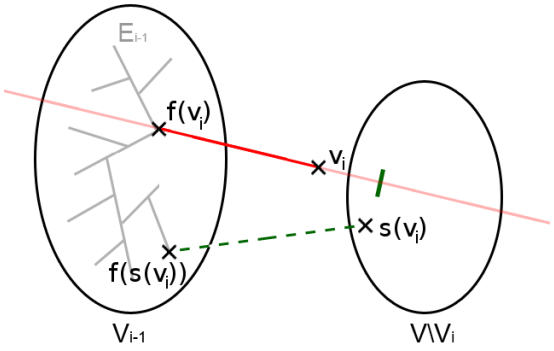
Ainsi, $\omega(f(s(v_i)), s(v_i))$ est la première distance candidate à la distance limite maximale. Nous la notons d_i^1 :

$$d_i^1 = \omega(s(v_i), (f \circ s)(v_i)). \tag{7}$$

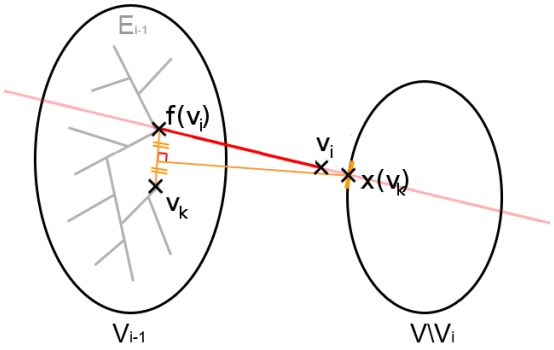
Nous devons avoir $f(v_i) = f(v^*)$ pour considérer que notre EMST est stable, $f(v_i)$ doit rester le sommet le plus proche de v^*. Donc pour chaque sommet $v_k \in V_{i-1}$ nous calculons $x(v_k) \in \mathbb{R}^3$ l'intersection entre la demi-droite $]f(v_i); v_i)$ et la médiatrice du segment $[f(v_i); v_k]$ illustrée en Figure 3.b.

Il est évident que nous avons : $d(f(v_i), x(v_k)) = d(v_k, x(v_k))$ et si $\omega(f(v_i), v^*) > d(f(v_i), x(v_k))$ alors v_i^* est plus proche de v_k que de $f(v_i)$. Donc, v^* doit vérifier, pour chaque $v_k \in V_{i-1}$ $\omega(f(v_i), v^*) < d(f(v_i), x(v_k))$. Ainsi, l'autre candidat à la distance limite maximum, notée d_i^2 s'exprime de la façon suivante :

$$d_i^2 = \min_{v_k \in V_{i-1}} \omega(p(v_i), x(v_k)). \tag{8}$$



(a)



(b)

Figure 3: *Schéma du calcul de la distance limite maximum (d^+) : a) Recheche du second sommet le plus proche de V_{i-1}, b) Intersection entre les médiatrices et la demi-droite $]f(v_i), v_i)$.*

Ces deux calculs de limite maximum ont pour but de sélectionner v^* à l'étape i de l'agorithme de Prim et de ne pas changer de "père", c'est-à-dire $f(v_i) = f(v^*)$. Nous devons vérifier ces deux conditions, nous prenons le minimum de ces deux distances :

$$d_i^+ = \min\{d_i^1, d_i^2\}. \tag{9}$$

Afin de ne pas modifier l'EMST à l'étape i, v^* doit vérifier :

$$\omega(f(v^*), v^*) < d_i^+. \tag{10}$$

L'équation (10) est démontrée en section B.

4. Resultats expérimentaux

Dans cette section, nous proposons une analyse statistique des distances limites minimum et maximum présentées en section 3.2. En section 4.1 nous décrirons les conditions expérimentales, puis en section 4.2 nous présentons un objet 3D en exemple pour faire un commentaire complet de nos résultats. Nous introduisons ensuite en section 4.3 des résultats de visualisation de zones robustes des EMST. Nous terminons en section 4.4 sur un comparatif entre deux normalisations et leurs impacts sur les mesures.

4.1. Conditions expérimentales

Pour la phase expérimentale, nous avons utilisé une base de données composée de 14 objets 3D issus de différentes sources (Stanford University Graphics Laboratory[†], le projet MADRAS[‡], Strategies S.A.[§] et Aimasharpe[¶]). Nous constatons, Figure 4, que ces objets sont de forme variée et sont utilisés dans de nombreux champs d'application tels que la CAO, la médecine, la manufacture et le divertissement.

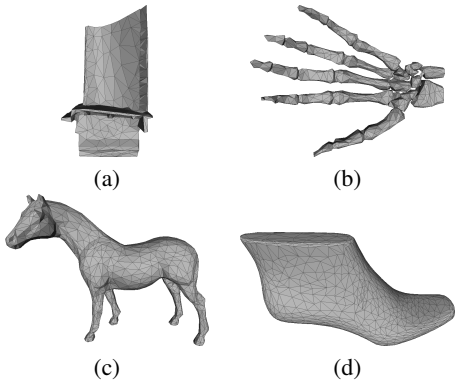


Figure 4: *Sélection d'objet 3D : a) Blade, b) Skeleton, c) Horse, d) Shoe.*

Afin de comparer ces maillages, nous les avons sous-échantillonnés de manière à avoir un nombre de sommets approximativement identique pour chacun d'eux. Pour des raisons pratiques nous prenons des objets d'environ 1000 sommets.

Pour pouvoir comparer les distributions des distances de déplacement, nous normalisons les objets avec deux méthodes :

1. la taille de la boîte englobante (normalisation (1));
2. la distance moyenne entre deux sommets (normalisation (2)).

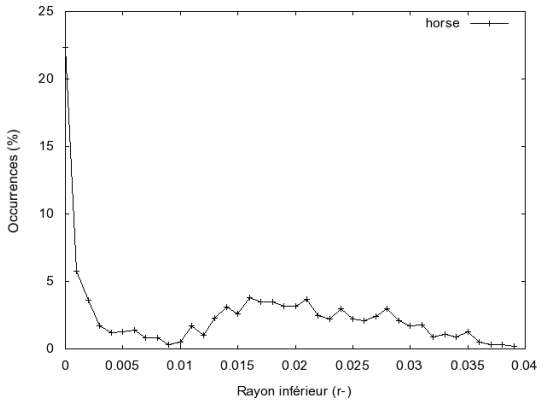
Puis en chaque sommet, nous calculons le rayon supérieur r_i^+ et le rayon inférieur r_i^- :

$$r_i^+ = d_i^+ - \omega(f(v_i), v_i); \tag{11}$$

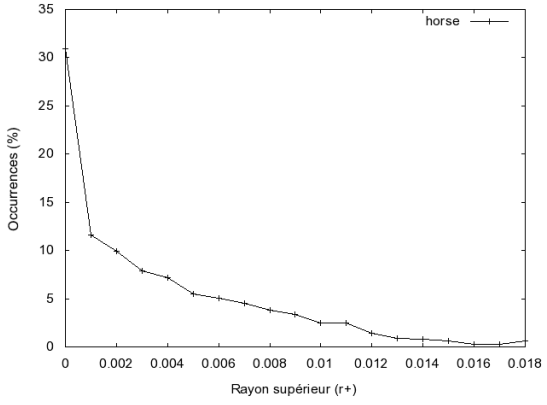
$$r_i^- = \omega(f(v_i), v_i) - d_i^-. \tag{12}$$

4.2. Un exemple complet : *Horse*

Prenons en exemple le maillage *Horse* illustré Figure 6.a normalisé en fonction de la taille de la boîte englobante. Les résultats des distributions de r^+ et r^- sont illustrés respectivement en Figure 5.a et 5.b.



(a)



(b)

Figure 5: *a) Distribution de r^-, b) Distribution de r^+ (pas d'échantillonage : 0.001).*

Pour chaque rayon calculé, plus la grandeur est importante, plus le sommet est robuste. Autrement dit, plus

[†] http://www-graphics.stanford.edu/
[‡] http://www-rech.telecom-lille1.eu/madras/
[§] http://www.cadwin.com/
[¶] http://dsw.aimatshape.net/

r^+ (respectivement r^-) est grand, plus le sommet v_i peut s'éloigner (respectivement se rapprocher) du "père" $f(v_i)$. Nous remarquons qu'un grand nombre de sommets ne peut absolument pas être déplacé soit en se rapprochant du "père" (22%), soit en s'éloignant (30%). Ces résultats ne sont pas étonnants, ils révèlent la fragilité de la structure.

Pour le rayon supérieur r^+, le nombre de sommets décroit rapidement lorsque le rayon augmente. La grande majorité des sommets ne peut pas être éloignée. De plus, notons que pour les sommets les plus robustes, le maximum des rayons inférieurs r^- est deux fois supérieur à celui des rayons supérieurs r^+. Ce qui montre que le déplacement du sommet vers son "père" est privilégié.

Pour le rayon inférieur r^-, nous remarquons que la courbe décroît brutalement, puis croît jusqu'à atteindre un maximum local autour d'une valeur comprise entre $0,015$ et $0,02$. Ce maximum local est très intéressant. Comme nous l'avons dit précédement, le déplacement vers le "père" est favorisé de telle sorte qu'on peut déplacer fortement certains sommets.

Cette première analyse nous permet de considérer que, dans les hypothèses de travail, soit on ne peut pas (ou très peu) déplacer le sommet, soit on peut le rapprocher de son "père" et dans certain cas très fortement.

4.3. Visualisation de zones robustes

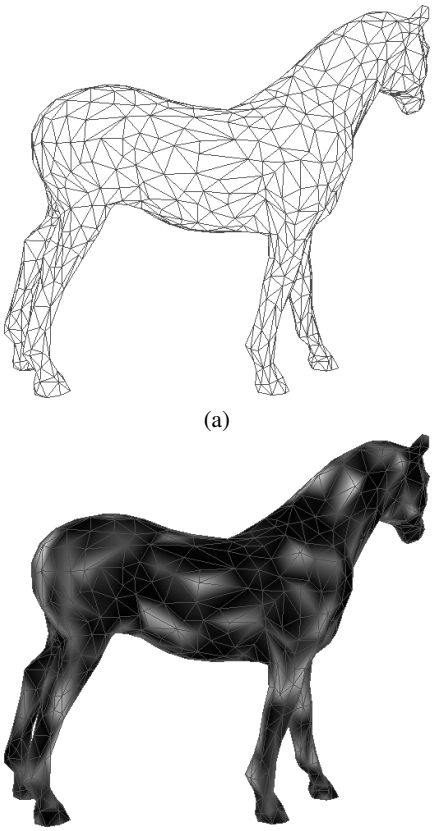
A la vue des résultats précédents, nous proposons de visualiser les zones les plus robustes d'un maillage lors d'une synchronisation par calcul d'EMST. Nous avons choisi de représenter Figure 6.b les sommets les plus robustes pour le rayon supérieur r_i^+. Plus les couleurs sont claires, plus les zones sont robustes car r^+ est grand.

Sans surprise, nous nous apercevons que beaucoup de zones sont considérées comme fragiles (zones sombres de l'image). Nous remarquons que pour la plupart des sommets robustes, les distances entre lui-même et ses voisins immédiats dans le maillage sont très variables, contrairement à la plupart des sommets fragiles où l'espacement entre lui-même et ses voisins est quasi-régulier. Ceci pourrait être une piste en vue d'un critère de robustesse géométrique.

4.4. Influence de la normalisation

Nous nous intéressons ici à l'influence de la normalisation. La Figure 7 présente les distributions pour le rayon inférieur r^- et pour tous les objets de notre base de données. La courbe de la Figure 7.a correspond aux maillages normalisés par la normalisation (1) et la Figure 7.b par la normalisation (2).

L'allure des tracés est globalement la même. Pour la normalisation (2), les maxima locaux sont détectés autour d'une même valeur. Pour la normalisation (1), ces maxima locaux ne sont pas positionnés autour de la même valeur, ce qui suggère que la distance moyenne entre deux sommets est fonction de la distance de rapprochement. En effet, nous savons qu'il existe certains sommets v_i que l'on peut fortement rapprocher de leur "père" $f(v_i)$. De plus, nous savons qu'il y a plus de chance pour un sommet v_i de se rapprocher que de s'éloigner de son "père" $f(v_i)$. Ce qui explique cette observation.

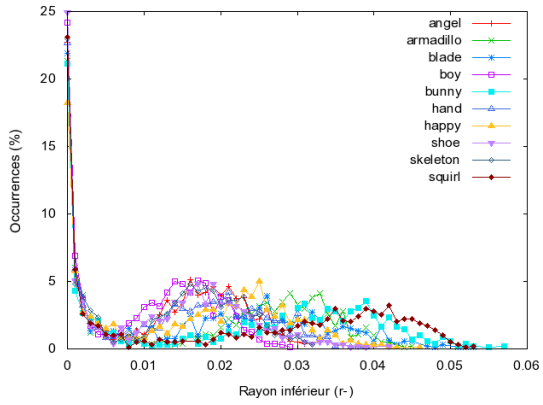


(a) (b)

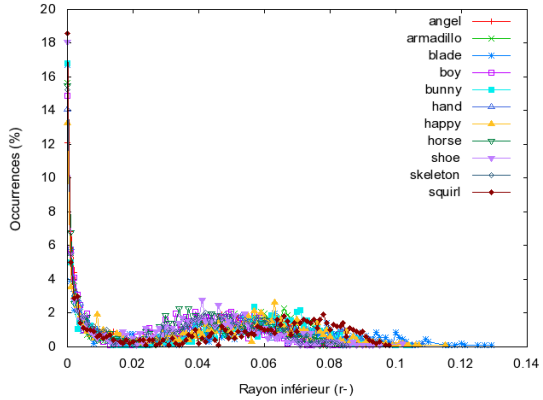
Figure 6: *Visualisation de* Horse*: a) Vue de face, b) Zones robustes en vue de face.*

Ainsi, suivant les applications et notre domaine d'étude, une normalisation peut être préférable à une autre. Ici, la normalisation (1) discrimine plus facilement les objets et permet de les classer plus facilement par niveau de robustesse.

Enfin, la Figure 8 illustre les distributions des rayons supérieurs r^+, pour les objets normalisés par la normalisation (1) en Figure 8.a et pour la normalisation (2) en Figure 8.b. Outre le fait que les valeurs maximales soient différentes, ceci étant dû à la différence des normalisations, les allures des courbes sont très similaires à un facteur multiplicatif près.

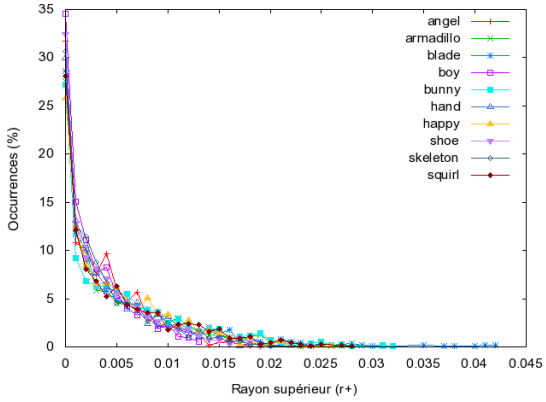


(a)

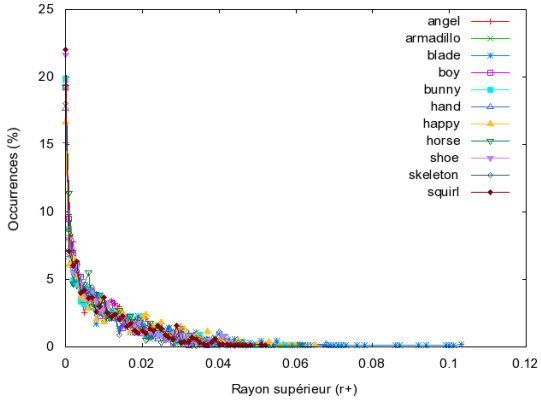


(b)

Figure 7: *Distribution de r^- pour la normalisation : a) En fonction de la taille de la boite englobante, b) En fonction de la distance moyenne entre deux sommets.*



(a)



(b)

Figure 8: *Distribution de r^+ pour la normalisation : a) En fonction de la taille de la boite englobante, b) En fonction de la distance moyenne entre deux sommets.*

5. Conclusion et perspectives

Dans cet article, nous avons présenté une nouvelle analyse de la sensibilité des EMST. Nous avons proposé des simplifications afin de rendre le problème plus abordable. Pour chaque sommet v_i, nous nous sommes intéressés au déplacement de ce sommet le long de la demi-droite $]f(v_i),v_i)$, avec $f(v_i)$, le "père" du sommet v_i obtenu par l'algorithme de Prim. Nous calculons pour chaque sommet v_i de combien il est possible de se rapprocher ou de s'éloigner de $f(v_i)$, et nous obtenons respectivement les distances r^- et r^+.

Cette méthode nous permet de visualiser les zones robustes d'un objet qui pouront être tatouées par synchronisation avec le calcul d'un EMST. L'objectif est d'améliorer cette analyse afin de prédire la robustesse du message au sein de l'objet 3D avant même de l'avoir insérer.

De plus, lorsqu'un objet 3D tatoué est transmis sur un canal bruité, certains sommets sont perturbés. C'est pourquoi nous nous intéresserons à la corrélation entre ces critères (r^+,r^-) et le déplacement des sommets subit par un bruit par exemple.

Appendix A: Démonstration : Calcul de d_i^-, distance de rapprochement limite

Rappelons l'hypothèse de notre problème simplifié, pour l'analyse de la stabilité des EMST : $\forall k\ k < i\ T_k = T_k^*$, les connexions dans l'arbre ne sont pas modifiées jusqu'à l'étape i. A l'étape i, nous souhaitons que :

1. $v^* = v_i^*$, le sommet déplacé à l'étape i est toujours sélectionné à cette étape après le déplacement;
2. $f(v_i) = f(v_i^*)$, le "père" du sommet sélectionné à l'étape i reste le même.

Nous devons démontrer que v^* doit vérifier la relation suivante pour conserver la même connexion à l'étape i dans l'EMST :

$$\omega(f(v^*),v^*) > d_i^- .$$

Etape 1

Soit $e_l = \{f(v_l), v_l\}$ l'arête telle que si elle existe $e_l = \max_{e_j \in E_{i-1} \setminus E_k} \omega(e_j)$. Si cette arête n'existe pas alors $E_{i-1} \setminus E_k$ est vide, donc $f(v_i) = v_{i-1}$, dans ce cas là le sommet v_i peut se déplacer aussi prêt que l'on souhaite de $f(v_i)$.

Supposons que $e_l = \{f(v_l), v_l\}$ existe alors $d_i^- = \omega(f(v_l), v_l)$.

Notons que $k \leq l < i$, donc l'algorithme de Prim ajoute dans l'arbre T^* d'abord v_k^*, puis plus tard v_l^* et à l'étape i qui nous intéresse v_i^*. Avec ces notations, démontrons par l'absurde que l'équation (4) doit être vérifiée pour que $v_i^* = v^*$.

Supposons $\omega(f(v^*), v^*) \leq \omega(f(v_l^*), v_l^*)$. En se plaçant à l'étape l de l'algorithme de Prim, on sait que $f(v_i^*) \in V_k \subset V_{l-1}$ et $f(v_l^*) \in V_{l-1}$ par définition. A l'étape l on choisit le sommet le plus proche de l'ensemble V_{l-1}, or $\omega(f(v^*), v^*) \leq \omega(f(v_l^*), v_l^*)$ donc $v^* = v_l^*$ ce qui est en contradiction avec notre hypothèse.

Etape 2

Il est évident que si v_i est le sommet le plus proche de $f(v_i)$ dans $V \setminus V_i$, si v^* est un sommet appartenant au segment $]f(v_i), v_i[$ alors il reste le plus proche sommet de $f(v_i)$ dans $V \setminus V_i$. Le "père" de v_i est le "père" de v^* qui est sélectionné à l'étape i. □

Appendix B: Démonstration : Calcul de d_i^+, distance d'éloignement limite

Nous devons démontrer que v^* doit vérifier la relation suivante pour conserver la même connexion à l'étape i dans l'EMST :

$$\omega(f(v^*), v^*) < d_i^+.$$

Etape 1

Tout d'abord montrons que $\omega(f(v^*), v^*) < d_i^1$.

Ceci est assez trivial, v_i étant le sommet le plus proche de V_{i-1} situé à distance $\omega(f(v_i), v_i)$ et $s(v_i)$ le deuxième sommet le plus proche de V_{i-1} situé à distance $\omega(f(s(v_i)), s(v_i))$. Supposons par l'absurde que $\omega(f(v^*), v^*) \geq d_i^1$, alors $s(v_i)$ est le sommet le plus proche de V_{i-1}, ce qui est contradictoire. □

Etape 2

Pour finir, montrons que $\omega(f(v^*), v^*) < d_i^2$.

Il est trivial que le long de la demi-droite $]f(v_i), v_i)$, les sommets $\{f(v_i), v_i, x(v_k)\}$ sont alignés dans cet ordre. De plus, $x(v_k)$ est le point à égale distance des sommets $f(v_i) \in V_{i-1}^*$ et $v_k \in V_{i-1}^*$. Ainsi $x(v_k)$ sépare la demi-droite $]v_i; \infty)$ en deux parties :

- $\forall v^* \in]v_i; x(v_k)[$, $\omega(f(v_i), v^*) < d(f(v_i), x(v_k))$, où v^* est plus proche de $f(v_i)$ que de v_k;
- $\forall v^* \in]x(v_k); \infty[$, $\omega(f(v_i), v^*) > d(f(v_i), x(v_k))$, où v^* est plus proche de v_k que de $f(v_i)$.

Ainsi, la propriété est démontrée. □

References

[APDP08] AMAT P., PUECH W., DRUON S., PEDEBOY J.: Lossless Data Hiding Method Based on MST and Topology Changes of 3D Triangular Mesh. In *EUSIPCO'08: 16th European Signal Processing Conference, Lausanne, Switzerland* (2008).

[DRT92] DIXON B., RAUCH M., TARJAN R.: Verification and Sensitivity Analysis of Minimum Spanning Trees in Linear Time. *SIAM Journal on Computing 21* (1992), 1184–1992.

[Gor99] GORDEEV E.: Stability Analysis of the Minimum Spanning Tree Problem. *Computational Mathematics and Mathematical Physics 39*, 5 (1999), 738–746.

[Gor01] GORDEEV E.: Stability Analysis in Optimization Problems on Matroids in the Metric l1. *Cybernetics and Systems Analysis 37* (2001), 251–259.

[Pri57] PRIM R.: Shortest Connection Networks and Some Generalizations. *Bell System Technical Journal 36* (1957), 1389–1401.

[WLDB08] WANG K., LAVOUÉ G., DENIS F., BASKURT A.: A Comprehensive Survey on Three-Dimensional Mesh Watermarking. *IEEE Transactions on Multimedia 10* (2008), 1513–1527.

Sensitivity Analysis of Euclidean Minimum Spanning Tree

N.Tournier[a, b], W. Puech[a], G. Subsol[a] and J-P. Pedeboy [b]

[a] LIRMM Laboratory, Univ. Montpellier II, CNRS
161 rue Ada, 34392 Montpellier, FRANCE;
[b] Strategies S.a, 41-43 rue de Villeneuve, Parc des Affaires SILIC - BP 80429
94583 Rungis, FRANCE

ABSTRACT

In 3D steganography and watermarking, the synchronization of the hidden data is a major problem. We need to know where the message is embedded in order to extract the correct information. Various algorithms have been proposed for the last couple of years and we focused on a method based on Euclidean minimum spanning tree (EMST) for the mesh vertices. In this paper we analyze the sensitivity of the EMST structure in order to propose a new method more robust. We present a new theoretical analysis and we propose to visualize the robustness of the EMST. Moreover, we can apply this analysis to various applications that can be useful in 3D steganography such fragile area detection and prediction of the 3D object robustness during transmission on a noisy channel.

Keywords: 3D steganography, watermarking, graph theory, Euclidean minimum spanning tree, sensitivity analysis

1. INTRODUCTION

Internet is very useful to broadcast multimedia information. There are more and more 3D object exchange in infography, CAO for example. It's important on the one hand, to protect the file content and on the other hand, to embed metadata. Data hiding may be a solution.

A recent survey of 3D watermarking techniques was proposed by Wang *et al.*[1] Let define a 3D object as a mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, with $\mathcal{V}$ the cloud of n vertices and $\mathcal{F}$ the set of facets. It is important to note, when we hide a message in the mesh we need to locate where the binary data is distributed. This is the synchronization process which is the main difficulty in 3D data hiding. Various methods have been proposed, and we interest in the synchronization by EMST computing proposed by Amat *et al.*[2]

An EMST is a minumum spanning tree (MST), a tree $T = (V, E_T)$ that joins all the vertices of a graph $G = (V, E)$ using the edges $e_i = \{v_s, v_t\} \in E$, with a weight $\omega(e_i) \in \mathbb{R}^+$, that minimize the total weight $\sum_{e_i \in E_T} \omega(e_i)$.

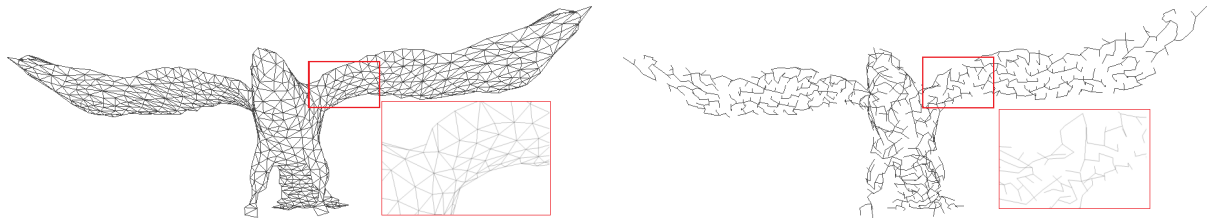
One of the original techniques to synchronize the message is proposed by Amat *et al.*[2] It is a method that does not move any vertex of the mesh, and they propose to use an EMST to be able to cover the mesh in a unique manner for the synchronization of the message. We can see Figure 1.a a representation of a mesh and its corresponding EMST Figure 1.b.

To embed a message into the mesh, they analyze for quadruples in the EMST that verify different conditions of planarity, convexity and overlapping. These quadruples are used for the data hiding and they are detected after the watermarking to extract the message. But this method is very fragile, so we are interested in the analysis of the sensitivity of EMST in order to locate the most robust vertices in the mesh.

As the method is fragile, the sensitivity analysis of EMST may be a solution to find a method more robust. We propose to compute how a vertex can be moved without changing the connexions in the EMST. Since the problem is very difficult, to make it tractable we propose to make some simplifications.

In this paper, we expose in Section 2 previous work on sensitivity analysis of minimum spanning tree (MST). In Section 3 we propose a theoretical sensitivity analysis of EMST for geometrical data. Then, in Section 4 we present experimental results of our approach. To conclude, in Section 5, we discuss about our future prospects.

Further author information:
E-mail: {nicolas.tournier, william.puech, gerard.subsol}@lirmm.fr, Telephone: +33 (0)4 67 41 85 85



(a) (b)

Figure 1. 3D representations of: a) A 3D Mesh, b) Its corresponding EMST computed by the Prim's algorithm.[3]

2. PREVIOUS WORK

According to Gordeev[4,5] the sensitivity analysis of minimum spanning tree (MST) is an optimization problem based on matroids. Let $G = (V, E)$ a graph with n nodes and m edges. Gordeev considers this following model. He lets $D_m = \{\tau_1, \ldots, \tau_q\}$ with $(q > 1)$ a system of subsets of E that he calls trajectories; $A = (a_1, \ldots, a_m) \subset \mathbb{R}^m$ such as $\forall i \ a_i = \omega(e_i)$, the weight of the edges of the graph G and $\tau(A)$ a functional called the trajectory length for A such as $\tau(A) = \sum_{e_i \in \tau} a_i$.

Therefore, the combinatory problem is defined with the pair (E, D_m), A is the variable to optimize in order to minimize the functional $\tau(A)$. With this model, Gordeev modelizes the MST problem with D_m the set of all spanning trees of G in which the MST is a trajectory that minimizes the functional $\tau(A)$.

Let $\psi(A)$ the indice set i of the optimal trajectories τ_i of the problem for a given A, and $B \in \mathbb{R}^m$ such as $\|B\| < \epsilon$ a perturbation vector. Gordeev talks about ϵ-stability when $\psi(A + B) \subset \psi(A)$.

In the MST problem, for a given noise intensity ϵ it exists some MST that are always solution of the MST problem after the perturbation. He deduces a stability radius $\rho(A) = \sup \epsilon$ such as A is ϵ-stable for the problem. Its algorithm is polynomial and its complexity is $O(n^3 m \log(\frac{n^2}{m}))$ and so for a complete graph $O(n^5))$.

In Dixon *et al.*,[6] the sensitivity analysis problem is to calculate, $\forall e_i \in E$ by how much $\omega(e_i)$ can change without affecting the minimality of the MST $T = (V, E_T)$. They divide the problem in two parts.

First, $\forall e_i \notin E_T$, they compute how much they can decrease $\omega(e_i)$ without changing the MST and, for all the edges $e_i \in E_T$, they compute how much they can raise the weight of $\omega(e_i)$ without changing the MST with a linear time complexity as a function of the number of edges.

In this Section, we have exposed two approaches to analyze the sensitivity of MST. We remind us of our data are geometrical, so:

- $G = (V, E)$ is a complete graph in the sense that each vertex is linked to all the others. This implies $m = n(n-1)$ and then maximizes the complexity of classical graph processing algorithms;
- the $\omega(e_i)$ are not independent. If $\omega(e_i)$ is pertubated, it implies that at least one of the two vertices of $e_i = \{v_s, v_t\}$ is perturbated and then that the $(n-2)$ edges coming from this vertex are perturbated.

As for Dixon *et al.* and Gordeev, the weights $\omega(e_i)$ are independant, that is why their propositions are not applicable in our case. So we propose a new approach in order to analyse the sensitivity of EMST based on geometrical data.

3. THEORETICAL SENSITIVITY ANALYSIS

Let a weighted graph $G = (V, E)$, with V the set of n nodes and E the set of edge such as:

- Each vertex is a node $v_i \in V$;
- $\forall v_s, v_t \in V \ ; \ v_s \neq v_t \ ; \ \exists e_i = \{v_s, v_t\} \in E$ and $\omega(e_i) = d(v_s, v_t)$, the Euclidean distance between the two vertices in the mesh.

3.1 Minimum spanning tree problem: reminders and notations

For a given connected weighted graph $G = (V, E)$, a representation of a 3D mesh $\mathcal{M}$, we propose to compute the EMST $T = (V, E_T)$. The MST computing is a well known polynomial problem. We choose to use the Prim's algorithm[3] because it is an incremental algorithm, and at each step we have a tree and a stable.

The algorithm starts from a given node $v_0 \in V$. At the i^{th} step, it adds to the current tree the closest node v_i between the tree and the remaining vertices. Thanks to this remark, we build a tree sequence $(T_i)_{0<i<n} = (V_i, E_i)_{0<i<n}$ such as $(V_i)_{0<i<n}$ is a sequence of node set and $(E_i)_{0<i<n}$ a sequence of edge set:

- $V_0 = \{v_0\}$;
- $E_0 = \emptyset$;
- $V_i = V_{i-1} \cup \{v_i\}$ $(\forall i < n)$, with $v_i \in \overline{V_{i-1}} = V \backslash V_{i-1}$ the closest vertex of V_{i-1}, defined in equation (1) ;
- $E_i = E_{i-1} \cup \{v_i, f(v_i)\}$ $(\forall i < n)$, with $f : V_i \to V_{i-1}$ a function that compute the closest node of v_i in V_{i-1} defined in equation (2).

$$v_i = \underset{v_j \in \overline{V_{i-1}}}{\text{Argmin}} \min_{v_k \in V_{i-1}} \omega(v_j, v_k); \tag{1}$$

$$f(v_i) = \begin{cases} \underset{v_j \in V_{i-1}}{\text{Argmin}} \min_{v_k \in \overline{V_{i-1}}} \omega(v_j, v_k), & i > 0 \\ v_0, & i = 0. \end{cases} \tag{2}$$

With these notations, the ESMT of G is $T = T_{n-1}$ and we propose, in this section, to study its sensitivity. In other words, we want to know how we can move a vertex in space such as the EMST of the graph representation does not change. It is a difficult problem because the graph G stands on a geometrical data. To make this problem tractable we propose to make some simplifications.

3.2 Sensitivity analysis problem and simplifications

Given a graph $G = (V, E)$, we compute its corresponding EMST at the step i of the Prim's algorithm $T_i = (V_i, E_i)$. We move the last vertex v_i in the 3D space, and denote the new vertex v^*. The graph G is then modified, and noted G^* the new one. Moreover, $\forall i$ $T_i^* = (V_i^*, E_i^*)$ the EMST computed by the Prim's algorithm at the step i.

As it is too complex to analyze the sensitivity of the EMST with respect to disruptions applied to all the vertices at the same time, we make the following simplification assumptions on the disruption of the vertices to make the problem tractable:

- At the step $i > 0$, we will disturb only the position of the vertex v_i, resulting in the disturbed vertex v^*. Moreover, we denote by G^* the new resulting graph and its EMST $T^* = (V^*, E^*)$;
- This geometric disruption will be restricted only to be along the half-line $]f(v_i), v_i)$;

Along the half-line $]f(v_i), v_i)$, we want to know how the vertex v_i can come up to $f(v_i)$ and how can it move away from $f(v_i)$ without changing the EMST at the step i.

With the previous notations, we suppose that T_i^* verify: $\forall k$, $k < i$; $T_k = T_k^*$. In other words, the connexions in the EMST are not modified until the step $(i-1)$. At the step i, we want:

1. $v^* = v_i^*$, the vertex selected at the step i is always selected at this step after the disruption;
2. $f(v_i) = f(v_i^*)$, the 'father' of the vertex selected at the step i is the same.

3.2.1 Minimum distance limit

Let $f(v_i)$, the closest vertex of v_i in V_{i-1}. Let V_k $(k < i)$ the smallest vertex set such as $f(v_i) \in V_k$ and $f(v_i) \notin V_{k-1}$ (see Figure 2). By composition, thanks to the Prim's algorithm, we affirm that all the edges of the EMST $e_j \in E_{i-1} \backslash E_k$ have a weight $\omega(e_j)$ less than $\omega(f(v_i), v_i)$ (equation (3)). The demonstration is presented in Section A.1.

$$\forall e_j = \{v_s, v_t\} \in E_{i-1} \backslash E_k; \omega(e_j) < \omega(f(v_i), v_i). \tag{3}$$

Therefore, we deduce the minimum distance limit d_i^- between v_i and $f(v_i)$:

$$d_i^- = \max_{e_j \in E_{i-1} \backslash E_k} \omega(e_j). \tag{4}$$

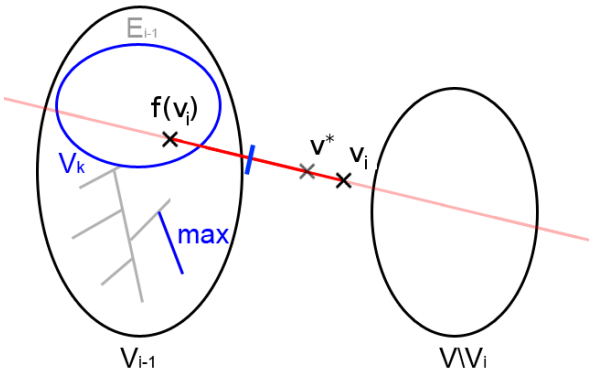


Figure 2. Scheme of the minimum limit computing.

So, to keep the same EMST, v^* must verify:

$$\omega(f(v^*), v^*) > d_i^-. \tag{5}$$

3.2.2 Maximum distance limit

The computing of the maximum distance limit is divided in two parts. First, in order to select the vertex v^* at the step i in the Prim's algorithm, we are looking for the second closest vertex of V_{i-1} denoted by $s(v_i)$ (equation (6)). Obviously, if v_i moves away from $f(v_i)$, it can be shift too far, otherwise it will be $s(v_i)$ the closest vertex of V_{i-1}. This situation is illustrated Figure 3.a.

$$s : V_i \to \overline{V_i}.$$

$$s(v_i) = \begin{cases} \underset{v_j \in V \backslash V_i}{\text{Argmin}} \ \underset{v_k \in V_{i-1}}{\min} \omega(v_j, v_k), & i < n-1 \\ v_{n-1}, & i = n-1. \end{cases} \tag{6}$$

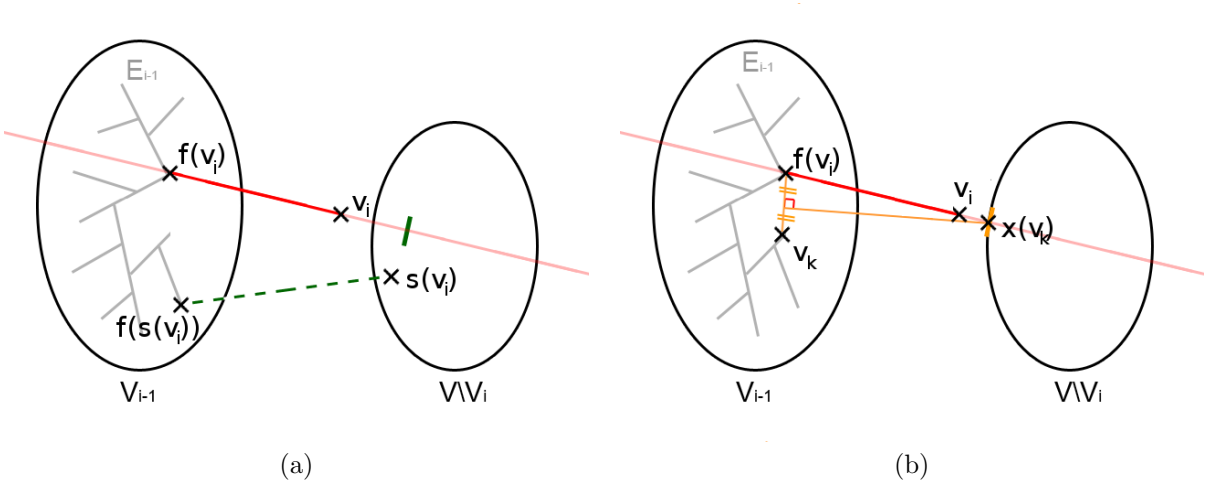
We need to know the distance between $s(v_i)$ and V_{i-1}, that is given by the edge $\{f(s(v_i)), s(v_i)\}$. Thus, we have a first candidate to the maximum distance computing, denotes by d_i^1:

$$d_i^1 = \omega(s(v_i), (f \circ s)(v_i)). \tag{7}$$

Secondly, we need to have $f(v_i) = f(v^*)$, so the vertex $f(v_i)$ must be the closest vertex of v^*. That is why for each vertex $v_k \in V_{i-1}$ we compute the intersection $x(v_k) \in \mathbb{R}^3$ between the half-line $]f(v_i), v_i)$ and the perpendicular bisector of the segment $[f(v_i), v_k]$, illustrated Figure 3.b.

Obviously, we have: $d(f(v_i), x(v_k)) = d(v_k, x(v_k))$ and if $\omega(f(v_i), v^*) > d(f(v_i), x(v_k))$ so v_i^* is closer from v_k than $f(v_i)$. So, v^* must verify for each $v_k \in V_{i-1}$ $\omega(f(v_i), v^*) < d(f(v_i), x(v_k))$. The other candidate to the maximum distance computing, denoted by d_i^2:

$$d_i^2 = \min_{v_k \in V_{i-1}} \omega(p(v_i), x(v_k)). \tag{8}$$



(a) (b)

Figure 3. Scheme of the maximum limit computing: a) Research of the second closest vertex, b) Intersection between perpendicular bisectors and the half-line $]f(v_i), v_i)$.

We compute two limit distances, one to select v^* at the step i of the Prim's algorithm and the other to $f(v_i) = f(v^*)$. We need to verify these two conditions, so the maximum limit distance is the minimum between d_i^1 and d_i^2, and v^* must verify the equation (10). The demonstration is described in Section A.2.

$$d_i^+ = \min\{d_i^1, d_i^2\}. \tag{9}$$

$$\omega(f(v^*), v^*) < d_i^+. \tag{10}$$

4. EXPERIMENTAL RESULTS

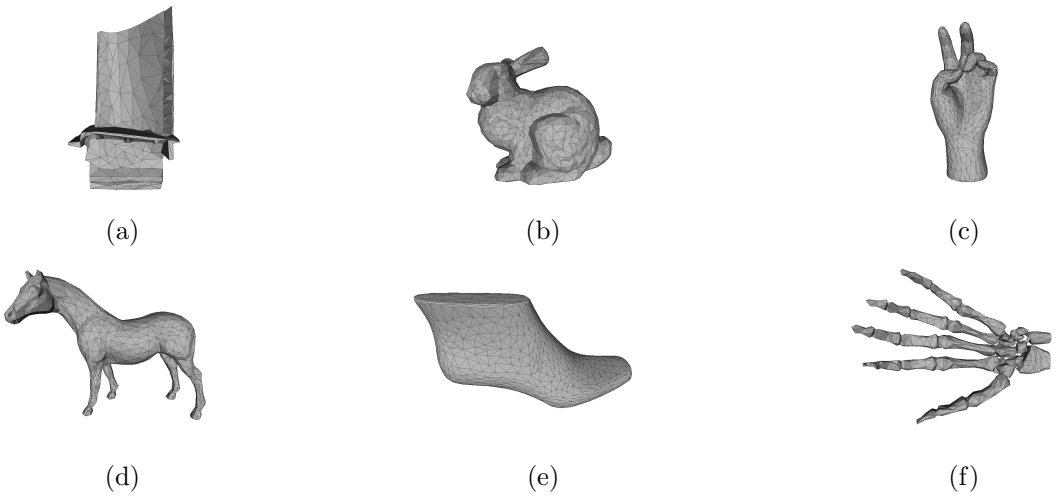
In this section, we propose to analyze statically the minimum and maximum distance limits presented in Section 3.2. In Section 4.1 we describe the experimental conditions, then we present a full an example with the 3D object *Angel*, illustrated in Figure 6.a and 6.c to comment the results and we introduce in Section 4.3 a first analysis by viewing the robust areas on the 3D object *Angel*. In Section 4.4, we apply two normalizations to compare the impact on our measures.

4.1 Experimental conditions

For the experiments we have used a database composed of 14 3D meshes selected from various sources (Stanford University Graphics Laboratory*, MADRAS project†, Strategies S.a‡ and Aimasharpe§). Their shapes are very

*http://www-graphics.stanford.edu
†http://www-rech.telecom-lille1.eu/madras
‡http://www.cadwin.com
§http://dsw.aimatshape.net

different as we can see on Figure 4 and they are used in different application field such CAD, manufactory, medicine or entertainment.



(a) (b) (c)

(d) (e) (f)

Figure 4. Selection of 3D objects: a) Blade, b) Bunny, c) Hand, d) Horse, e) Shoe, f) Skeleton.

In order to compare the meshes, we have subsampled them to have approximately the same number of vertices. For practical reasons, we choose around 1,000 vertices.

To be able to compare their disruptions, we have normalized all these objects, using two normalizations based on:

1. The size of the bounding box (normalization (1));
2. The average distance between two vertices in the mesh (normalization (2)).

Then we compute for each normalized mesh and for each vertex, the upper radius r_i^+ and the lower one r_i^-:

$$r_i^+ = d_i^+ - \omega(f(v_i), v_i); \tag{11}$$

$$r_i^- = \omega(f(v_i), v_i) - d_i^-. \tag{12}$$

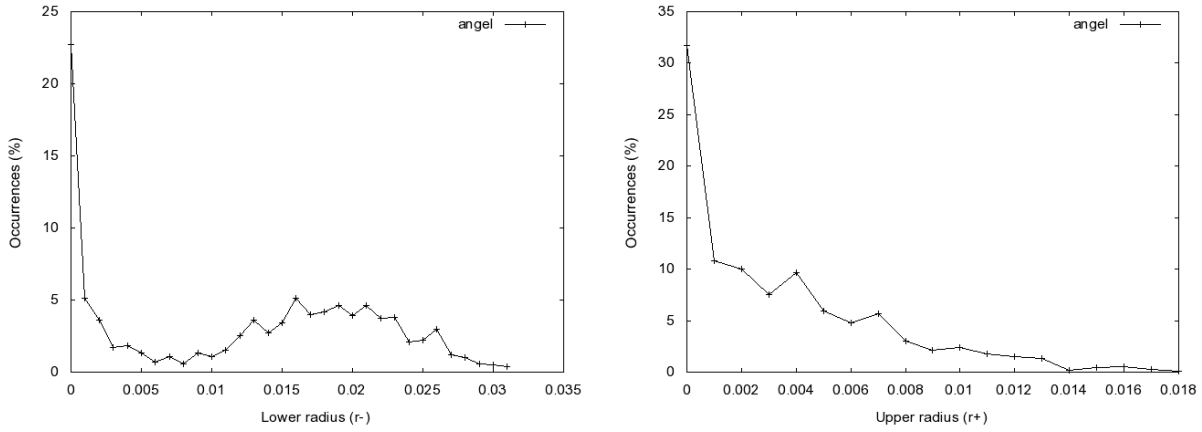
4.2 A full example based on *Angel*

The results of our analysis are presented for the mesh *Angel* normalized by the normalization (1) illustrated Figures 6.a. and 6.c. The Figures 5.a and 5.b represents respectively the distributions of r^+ and r^- subsampled each 0.001.

First, we remark there are many vertices that cannot be moved on one direction, more than 20% for the lower radius r^- and around 30% for the upper radius r^+. More the radius value is significant, more the vertex is robust. The results are not amazing; it reveals the fragility of the EMST structure.

For the upper radius r^+, the occurrences are decreasing very fast when the radius value increases. Lets note the maximum upper radius is around 0.018 whereas as for the lower radius the maximum is around 0.030. So it is easier to move a vertex v_i closer from its 'father' $f(v_i)$ than moving it away.

Then, for the lower radius, we can note an interesting peek between 0.015 and 0.02. With this observation, we can consider two kinds of vertex: those that cannot be moved closer from theirs 'father' and those can be moved almost as close as we want to.

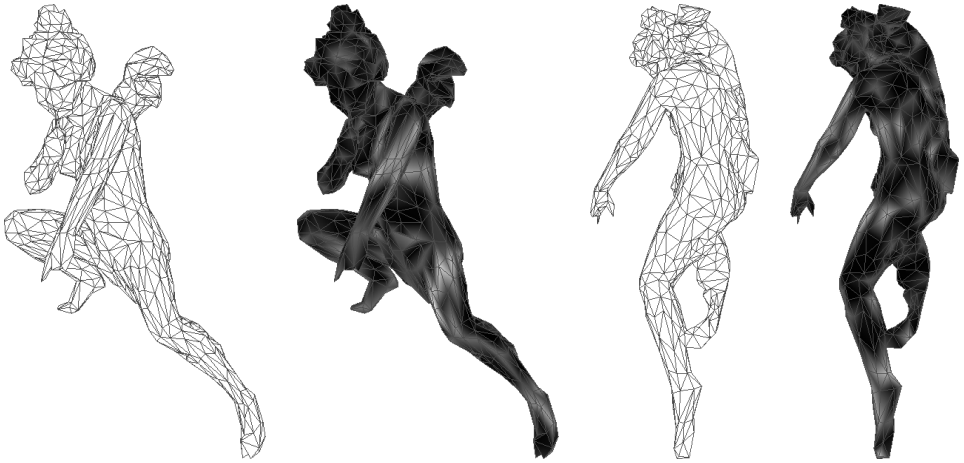


(a) (b)

Figure 5. Lower and upper radius for the mesh *Angel*: a) Distribution of the lower radius r^- (sampled each 0.001), b) Distribution of the upper radius r^+ (sampled each 0.001).

4.3 Robust area visualization

We propose to visualize the more robust vertices in order to locate the most robust areas to synchronize the hidden data during the watermarking process. To connect our theoretical analysis criteria defined by r^+ and r^-, for each vertex, we have choosen to use the upper radius r_i^+ to visualize these more robust areas. Figure 6.b and 6.d illustrate the result of the visualization. The darkest colors correspond to the most fragile areas, whereas the lightest colors are for the most robust areas.



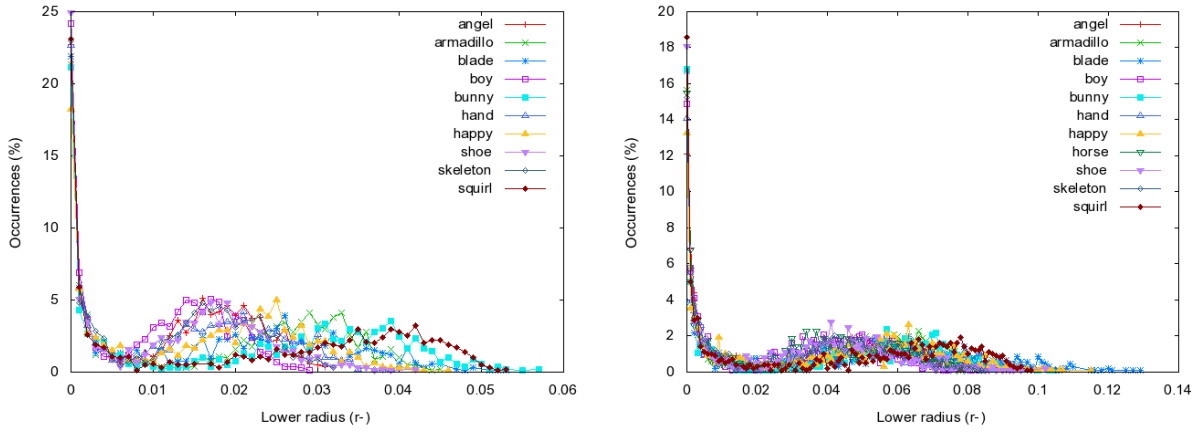
(a) (b) (c) (d)

Figure 6. Visualizations of *Angel*: a) Front view, b) Robust areas in front view, c) Behind view, d) Robust areas in behind view.

As we can see a lot of areas are fragile, illustrated in darkest color. And we can quote, for the most robust vertices, the spacing between itself and its neibourhood is not regular. It might be a lead for the research of geometrical criterion.

4.4 Impact of the normalization

Figure 7 illustrates the distributions of the lower radius r^- for the 3D objects of the database. On Figure 7.a, the objects are normalized by normalization (1) and on the Figure 7.b by the normalization (2).



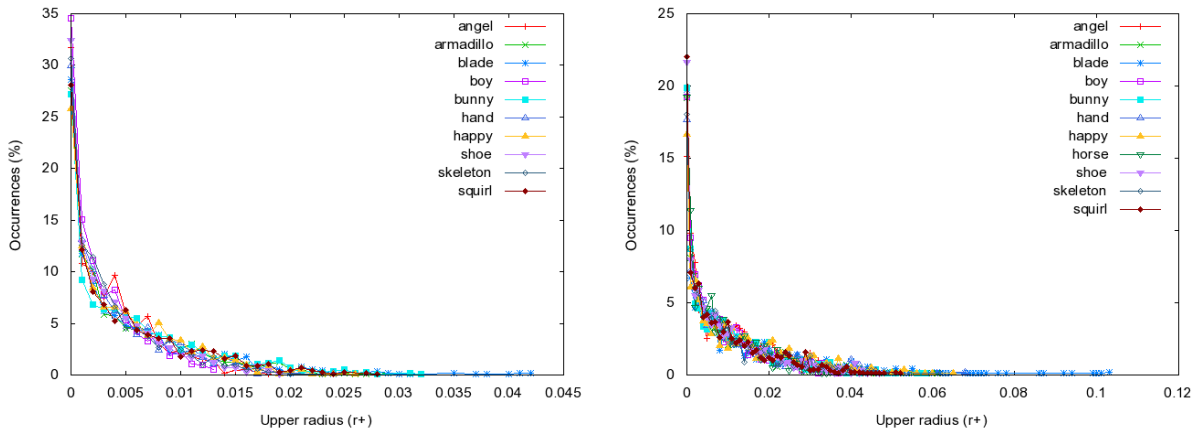
(a) (b)

Figure 7. Distribution of the lower radius (sampled each 0.001) for the normalization: a). As a function of the bounding box size, b) As a function of the average distance between two vertices in the mesh.

We remark the shape of the plots are globaly similar. For the normalization (2) the peak is almost at the same value. Otherwize, for the normalization (1), for each object, the peak is not positioned at the same value.

The normalization as a function of the size of the bounding box is more discriminatory for this criterion and it might be a robustness criterion.

Figure 8 shows the distributions of the upper radius r^+ for some 3D objects. As previously, Figure 8.a illustrates the results for the normalization (1) and Figure 8.b for the normalization (2).



(a) (b)

Figure 8. Distribution of the upper radius (sampled each 0.001) for the normalization: a). As a function of the bounding box size, b) as a function of the average distance between two vertices in the mesh.

Except the maximum upper radius for the normalization (2) that is 2 at 3 times upper the other normalization, the plots are also globaly similar.

5. CONCLUSION AND PERSPECTIVES

To conclude, we propose a new approach to analyse the sensitivity of the EMST based on geometrical data. We need to make some simplification in order to make the problem more treatable. For each vertex v_i, we consider it can move only along a streght half-line $]f(v_i), v_i)$, with $f(v_i)$, the 'father' of v_i in the EMST according to

the Prim's algorithm. And we compute how v_i can be closer and away from $f(v_i)$, these distances respectively correspond to the lower radius r^- and the upper radius r^+.

This method will permete us to visualize the robust areas of the object for a synchronisation by EMST computing. We expose our first results for given criteria. We have to perform this kind of appilication in order to predict the robustness of the watermarking technique.

Moreover, we are intereted in, what ahappen when we move all the vertices. For exemple, when a 3D object is transmitted on a noisy channel, some vertices are moved. So we need to make a correlation between our robust area prediction and the impact of a noise on the 3D object.

APPENDIX A. DEMONSTRATION

We suppose v_0 is always the same seed of the Prim's algorithm, and $\forall i\ V_{i-1} = V^*_{i-1}$. The proofs are decomposed in two parts:

1. $\forall i\ v^*$ is selected at the i^{th} step of the Prim's algorithm;
2. $\forall i\ f(v^*) = f(v_i)$.

A.1 Minimum limit computing

Referring to the equation (4), lets demonstrate this relation in order to keep the same EMST, according to the hypothesis of the sensitivity problem:

$$\omega(f(v^*), v^*) > d_i^- = \max_{e_j \in (V_{i-1} \setminus V_k)^2} \omega(e_j)$$

A.1.1

We denote by V_k the smallest set that contains the vertex $f(v_i)$ (i.e $f(v_i) \notin V_{k-1}$ and $f(v_i) \in V_k$); and $e_l = \{f(v_l), v_l\}$ the edge, if it exist, verifying $e_l = \max_{e_j \in (V_{i-1} \setminus V_k)^2} \omega(e_j)$. If this edge does not exists, it means $d_i^- = 0$ so we can move the vertex v_i as closed as we want from $f(v_i)$.

We suppose that it exists at least one edge e_l such as $e_l = \max_{e_j \in (V_{i-1} \setminus V_k)^2} \omega(e_j)$:

$$d_i^- = \omega(f(v_l), v_l).$$

It is important to note that $k \leq l < i$, in other words, in the chronogical vertex selection in the Prim's algorithm v_k is selected before v_l, and v_l before v_i. With this scheme, lets demonstrate *reductio ad absurdum* that the equation (4) must be verified to keep the same order in the sequence $(V_i^*)_{0<i<n}$.

Supposing $\omega(f(v^*), v^*) \leq d_i^- \Rightarrow \omega(f(v^*), v^*) \leq \omega(f(v_l^*), v_l^*)$. At the $(l-1)^{th}$ step of the Prim's algorithm we know $f(v_i^*), f(v_l^*) \in V_{l-1}$. According to the hypothesis $\omega(f(v^*), v^*) \leq \omega(f(v_l^*), v_l^*)$, so v^* will be chosen at the $(l-1)^{th}$. That is in contradiction to the stability of the EMST. □

A.1.2

Obviously, v_i is the closest node of $f(v_i)$ in $V \setminus V_i$, if we move v_i along the half-line $)f(v_i), v_i)$ closer from $f(v_i)$, the resulting vertex $v^* \in V \setminus V_i$ is the closest node of $f(v_i)$. In conclusion, the 'father' of v_i is also the 'father' of v^* with this displacement. □

A.2 Maximum limit computing

Referring to the equation (9), lets demonstrate this relation in order to keep the same EMST, according to the hypothesis of the sensitivity problem:

$$\omega(f(v^*), v^*) < d_i^+ = \min\{d_i^1, d_i^2\};$$

$$d_i^1 = \omega(s(v_i), (f \circ s)(v_i));$$

$$d_i^2 = \min_{v_k \in V_{i-1}} \omega(p(v_i), x(v_k)).$$

A.2.1

Lets demonstrate *reductio ad absurdum* that $\omega(f(v^*), v^*) < d_i^1$ must be verified to keep the same order in the sequence $(V_i^*)_{0<i<n}$.

We suppose $\omega(f(v^*), v^*) \geq d_i^1 = \omega(s(v_i), (f \circ s)(v_i))$. According to the hypothesis v_i is the closest vertex of V_{i-1}, and $s(v_i)$ the second one. Then, v_i is disturb in v^* but the other vertices does not move. Moreover $(f \circ s)(v_i), f(v^*) \in V_{i-1}$ and $s(v_i), v^* \in V \backslash V_{i-1}$. So according to the Prim's algorithm at the step i it choose the closest vertex of V_{i-1}, it is $s(v_i)$. v^* is to far from $f(v_i)$, so to verify the condition of our EMST stability problem $\omega(f(v^*), v^*) < d_i^1$. □

A.2.2

Let $v_k \in V_{i-1}$ a vertex satisfying the relation $\omega(v_k, x(v_k)) = \min_{v_j \in V_{i-1}} \omega(v_j, x(v_j))$. Obviously, on the line $(f(v_i), v_i)$, the vertices $\{f(v_i), v_i, x(v_k)\}$ are aligned in this order.

Moreover, $x(v_k)$ is the equidistant vertex between $f(v_i)$ and v_k. Obviously it separates the half-line $]v_i; \infty)$ in two parts:

- $\forall v^* \in]v_i; x(v_k)[$, $\omega(f(v_i), v^*) < d(f(v_i), x(v_k))$, v^* is closer from $f(v_i)$ than v_k;
- $\forall v^* \in]x(v_k); \infty[$, $\omega(f(v_i), v^*) > d(f(v_i), x(v_k))$, v^* is closer from v_k than $f(v_i)$.

It proves the proposition. □

REFERENCES

[1] Wang, K., Lavoué, G., Denis, F., and Baskurt, A., "A Comprehensive Survey on Three-Dimensional Mesh Watermarking," *IEEE Transactions on Multimedia* **10**, 1513–1527 (2008).

[2] Amat, P., Puech, W., Druon, S., and Pedeboy, J., "Lossless Data Hiding Method Based on MST and Topology Changes of 3D Triangular Mesh," in [*EUSIPCO'08: 16th European Signal Processing Conference, Lausanne, Switzerland*], (2008).

[3] Prim, R., "Shortest Connection Networks and Some Generalizations," *Bell System Technical Journal* **36**, 1389–1401 (1957).

[4] Gordeev, E., "Stability Analysis of the Minimum Spanning Tree Problem," *Computational Mathematics and Mathematical Physics* **39**(5), 738–746 (1999).

[5] Gordeev, E., "Stability Analysis in Optimization Problems on Matroids in the Metric l1," *Cybernetics and Systems Analysis* **37**, 251–259 (2001).

[6] Dixon, B., Rauch, M., and Tarjan, R., "Verification and Sensitivity Analysis of Minimum Spanning Trees in Linear Time," *SIAM Journal on Computing* **21**, 1184–1992 (1992).

3D Data Hiding for Enhancement and Indexation on Multimedia Medical Data

N. Tournier[1,2], G. Subsol[1], W. Puech[1], and J-P. Pedeboy[2]

[1] Université de Montpellier 2, LIRMM, CNRS,
161 rue Ada
Montpellier, France.

[2] STRATEGIES S.A.,
41-43 rue de Villemneuve
Parc des Affaires SILIC - BP 80429
Rungis, France.

Abstract. In medical applications, large quantities of multimedia data are exchanged such as 3D data acquired either by volume (CT-scan) or surface (laser range) scanning. The increasing of the numerical data using raises some unsolved issues. As for us, we are interested in the protection and the enhancement of multimedia content by insertion of hidden message. According to [7] some of these challenges are:

- Metadata embedding;
- Indexing and searching in database.

Data hiding may be a solution for these main applications in the medical domain. It is possible to embed metadata, with security for confidential data or for indexing area in a media, without increasing the size of the file.
In this article, we propose first to introduce briefly the data hiding principle in Section 1, we illustrate the principle with some previous works, and we present some useful applications for medical multimedia data. Then, in Section 2, we present our watermarked schemes on 3D images and, in Section 3, on 3D meshes. Finally, we conclude in Section 4.

1 Introduction

The main interest of data hiding is to embed additional information without increasing the file size and keeping the compatibility with the norms and the standards such as JPEG for image, DICOM for medical image, MPEG for video, etc.

1.1 Data hiding principle

The principle is to embed a message into the useful data of the multimedia file and to be able to extract or recognize it from the watermarked file. We can divide the insertion process in two parts: the synchronization and the embedding.

Synchronization allows us to know where the encoded message can be embedded in the host signal. It can be considered as a preparation of the media to the embedding. The host signal can be represented in other domain, such as a frequential domain (Discrete Cosines Transform, Fourier Transform), a wavelet domain (Discrete Wavelet Transform) as a function of the application and the chosen method. The aim is to be able to recognize the same subspace, called insertion subspace, before and after the watermarking.

The message is encoded as a function of a secret key and the host signal. The embedding consists in merging the encoded message and the host signal in the subspace domain. The embedding principle is illustrated on Fig. 1.

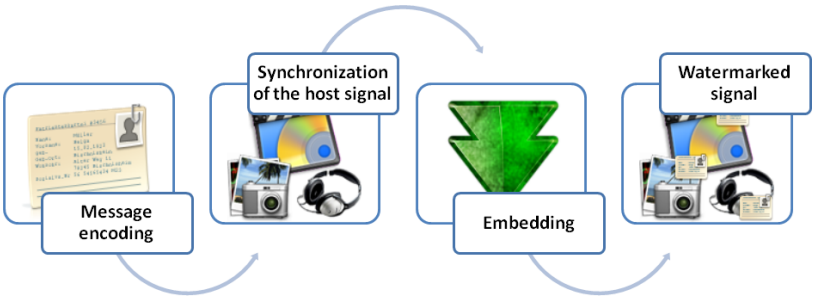


Fig. 1. Watermarking Scheme

On the watermarked data, we need to extract the embedded message. For the extraction process, first we synchronize the watermarked data to find the insertion subspace again and we extract the message depending on the insertion method. The extraction principle is illustrated on Fig. 2.

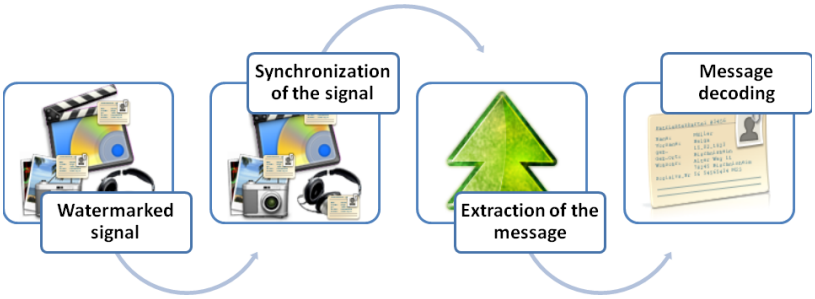


Fig. 2. Extraction Scheme

1.2 Data hiding applications in healthcare

The applications are various and depend on compromises between the capacity of the message to embed, the insertion robustness, the modification imperceptibility [11], the algorithm security [6] and the complexity. As the data are voluminous, the algorithms must be fast with a low complexity.

One of the main application of data hiding is communication, a transmission of a secret message. It looks like cryptography approaches. In data hiding we do not want to secure the media to transmit, the media is a carrier for the message we want to send. In this condition, the watermark algorithm must be robust and secured. For example, in a medical context, it could be interesting to embed the identity of the patient into the 3D image acquired by a CT-scan. For privacy, we need to keep anonymous its identity for a secured transmission of the data. We can note cryptography and data hiding can be combined for these applications [12].

Data hiding can be used for enhance the media with metadata. It is quite easy to add metadata in a file, but we do not want to increase the size of the media because more the data are voluminous; more it is difficult to store them. Such as the embedding of indexing information [3] in order to identify the media. It can be also possible to carry a diagnostic resume, a description of the image, some quotes or remarks in order to have all the main information available on the same file. But there are some constraints for example in ROI medical image, where the image must not be disrupt in the ROI areas [14].

For further information on data hiding application in healthcare, [5] made a review of image watermarking applications.

As we can see, data hiding is useful in medical application. Nowadays, there are a lot of well-known techniques in 2D watermarking. In the 3D domain, the topic is more difficult because the synchronization issue is more difficult. There is not any way to scan the 3D object obviously rather than in 1D scanning the audio with a pseudo-random clock; or in 2D scanning the image line by line for example.

In this section, we propose data hiding system for 3D objects. First, we introduce a watermark technique on 3D image acquired by CT-scan in Section 2 and in Section 3 we deal with a watermark technique applied on a 3D mesh. The presentation of the techniques is validated by experimental results in the respective sections.

2 3D image data hiding

2.1 2D data hiding on CT-scan before 3D reconstruction

For this approach, the idea is to use well-known 2D watermarking technique for 3D object. First, the object is acquired by volume (CT-scan) in order to have a 3D image. Each slice is equivalent to a 2D image that we can watermark by classical techniques. We illustrate on Fig. 3 some slices of a 3D image acquired by CT-scan.

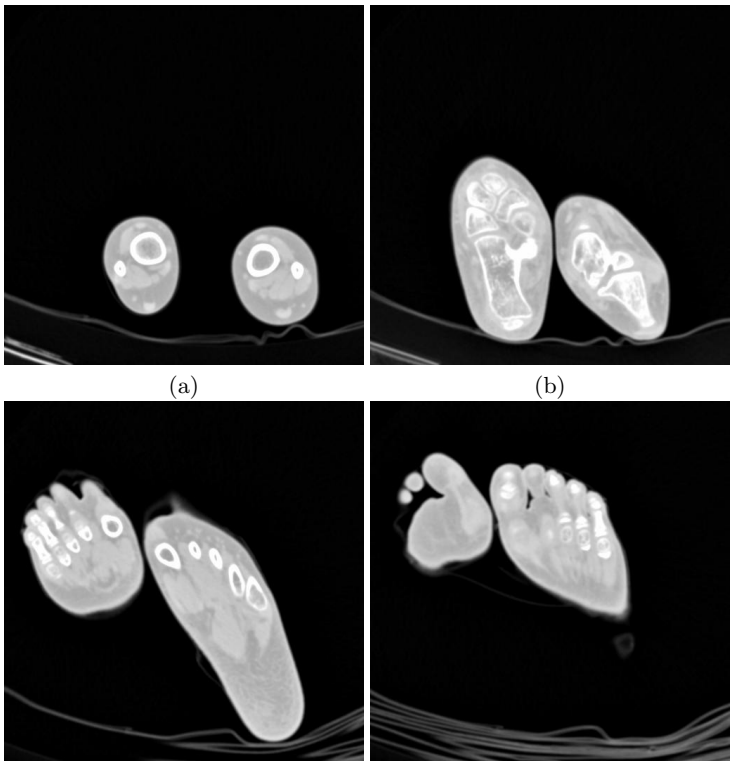


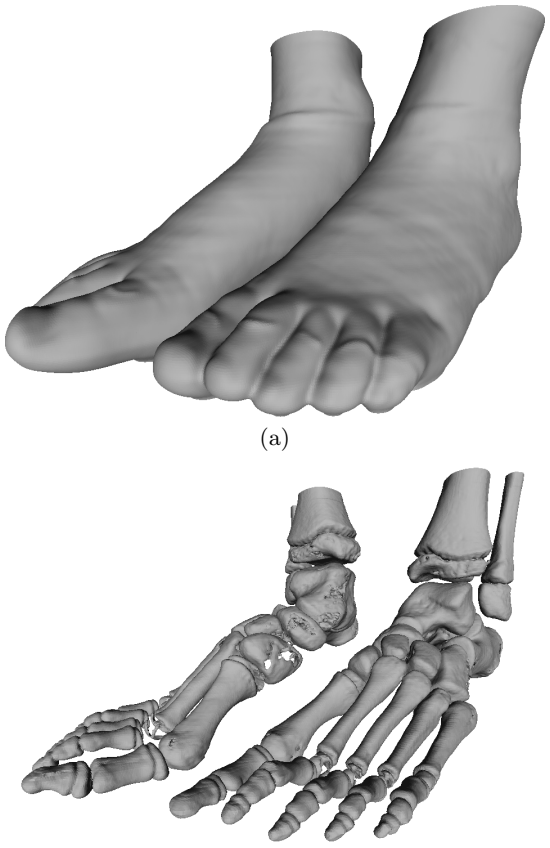
Fig. 3. An example of some 3D image slices: a) A represention of the slice #1, b. the slice #130, c. the slice #200 and d. the slice #230.

From the 3D image we are able to build a 3D mesh by computing the iso-surface. We illustrate the reconstruction of the skin (Fig. 4.a.) and the skeleton (Fig. 4.b.) from our 3D image. The aim is to watermark the 3D image such as there is no perceptual difference on the 3D object.

In our condition, we have 3D images[3] composed by 249 slices (512×512 pixels). We can watermark each slice with the same message, or independently with a different message for each one. It depends on the final application. For example, if we prefer to have a robust watermarking scheme, we will watermark the same message in each slice; else if we want to enhance the media with metadata, we prefer to have an high capacity watermarking scheme so we embed different message in each slice.

In that case, we just want to prove that the watermark keep a good quality on the 3D reconstruction, so it does not matter if the message is the same or

[3] http://pubimage.hcuge.ch:8080/.



(a)

(b)

Fig. 4. An example of 3D mesh reconstruction from the 3D image: a) The skin reconstruction composed by $663'072$ vertices and $1'126'466$ faces; b) The skeleton one composed by $804'073$ vertices and $1'577'853$ faces.)

not. We choose a well-known technique, by substitution of the LSB value of the pixels [2].

When the images are watermarked, we reconstruct the object from the 3D image into a mesh [8].

For the extraction procedure, we just need to extract the message from the 3D image by reading the LSB value of each watermarked pixel. It is a fragile technique with high capacity, fast (low complexity) and quite efficient to enhance 3D images with metadata.

2.2 Experimental results

From the 3D image composed by 249 slices (512×512 pixels) we embed in each slice 30ko. With the LSB watermarking method, associated with a pseudo-random image scanning; we can embed one bit per pixel. In these conditions, the capacity is fixed at 0.9265 bit per pixel.

First, we are looking for the quality of the watermark on each image. We compute the PSNR (Peak Signal Noise Rate) between the original slice and the watermark one. The results are very good, the PSNR equals to 51.63 dB in average for an high capacity application. There is no perceptual difference between a slice and the respective watermarked one (Fig. 5).

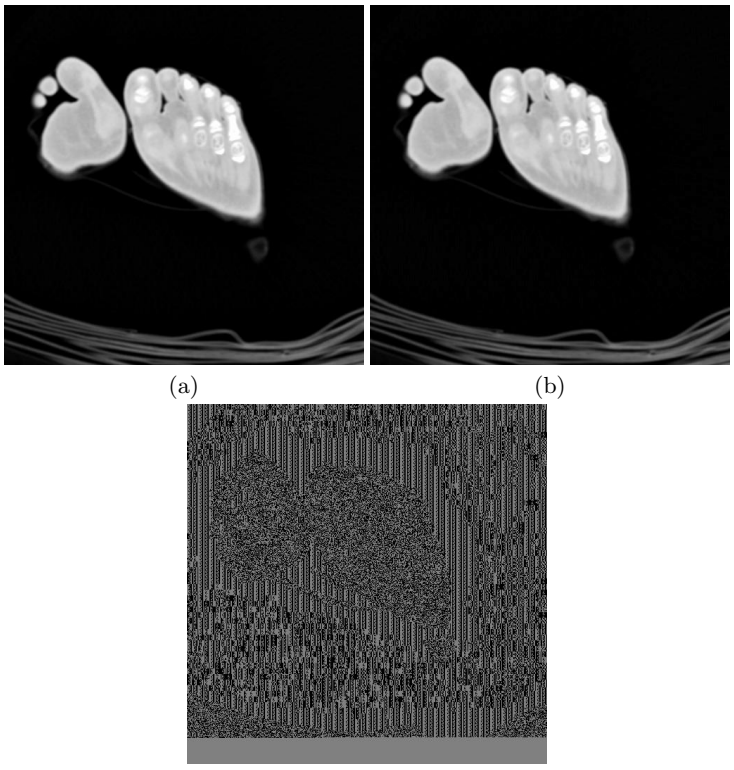


Fig. 5. Comparison between: a) The original slice #230; b) The watermarked image; c) The difference image.

The main aim is to know what is the impact of the watermark on the 3D mesh reconstruction. We build the 3D mesh, by computing iso-surface on the

watermarked 3D image and we compare the quality of the mesh obtained by the original image and this one.

To quantify the quality we use the Hausdorff distance, that computes the longest distance of the shortest distance between a vertex in the first mesh to a vertex in the second one. For the skeleton mesh and for the skin mesh, the Hausdorff distance equals to 0.00. That means there is no difference between these two meshes and we are able to extract the watermark from the 3D image. It is a very high capacity technique, nevertheless we can not extract the message from the 3D mesh.

3 3D mesh data hiding

3.1 3D data hiding using Euclidean Minimum Spanning Tree

In Section 2, we have presented a 2D watermark technique used for 3D objects. We remark the extraction process must be done from the 3D image and not from the 3D mesh. In this Section, we want to watermark the mesh. Various techniques exist, most of them are robust to geometrical rigid transformations. In this Section, we present an original 3D watermarking approach that does not move any vertex [1].

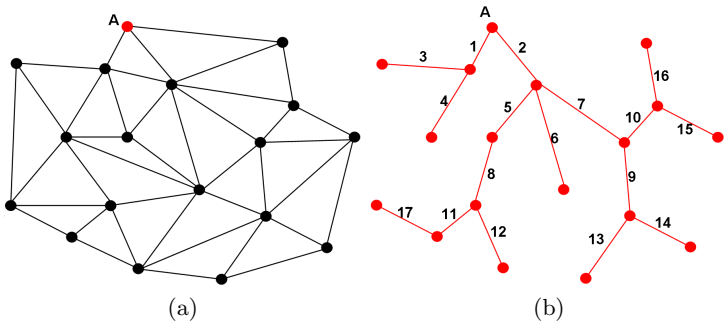


Fig. 6. An example of a mesh: a) The EMST of the mesh; b) We can remark connections in the EMST are not necessarily edges in the mesh.

Few 3D watermarking approaches do not move any vertex [1], [10], [9]. The originality of [1] is to watermark a message by editing the connections in the mesh without increasing the file size and without moving the vertices. In order to synchronize the message they compute an Euclidean minimum spanning tree (EMST) illustrated Fig. 6. The EMST is unique, and from a seed vertex v_0, the path of the vertices is also unique. This can be an interesting synchronization tool because we can cover the mesh with a unique path.

In the EMST, they are searching quadruples, a father-vertex and three sons, to embed one bit. The insertion is quite simple, if we want to embed a 0-bit the

edge of the EMST is chosen. Otherwise, if we want to embed a 1-bit, the other edge is in the EMST. The insertion process is illustrated Fig. 7.

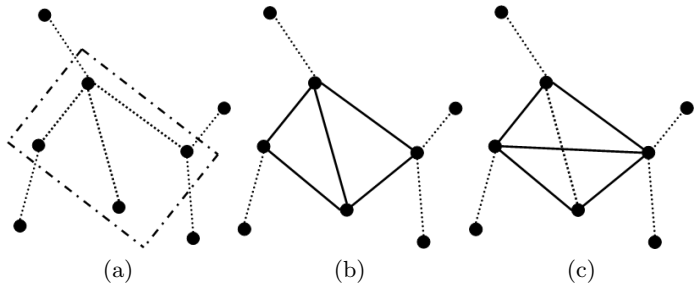


Fig. 7. Illustration of the EMST (a), the mesh when a 0-bit is embedded (b) and when a 1-bit (c).

In order to not create important visual distortions and synchronizations issues, the quadruples must verify the following conditions:

- Coplanarity: the measure of the angle formed by the two triangles (Fig. 8);
- Convexity: the quadruple must be convex in order to cover quite the same surface (Fig. 9);
- Covering: if two quadruples are neighbours only one of them is used for the embedding.

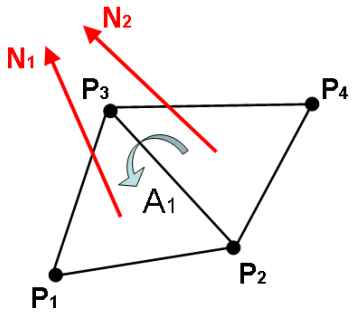
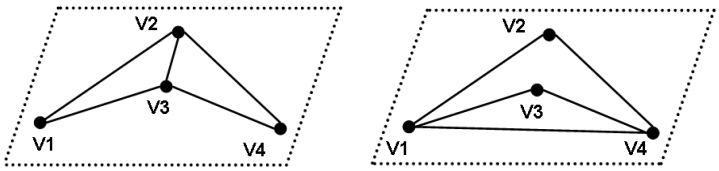


Fig. 8. Representation of the coplanarity condition.

3.2 Experimental results

We present some results on our 3D skeleton. We illustrate, Figure 10.a the simplified 3D mesh and its EMST Figure 10.b. During the watermarking process,



(a) (b)

Fig. 9. Representation of the watermarking impact on a quadruple not convex before the watermark (a) and after (b). The covered area is different and there is a topological problem: v_3 must not be inside of the triangle (v_1, v_2, v_4).

the vertices are not moving, the cloud is the same, only the connectivity is changed. Indeed, there is not any distortion due to the vertex displacement. So we compute the Hausdorff error, including the connectivity distortion. The Hausdorff distance is computed on the center of the triangles and not on the vertices and normalised as a function of the bounding box diagonal computed with METRO [4].

As a function of the coplanarity threshold, we can embed more information in the mesh. More the threshold is low, more the connectivity distortion is better but we select fewer quadruples than a high threshold.

For a coplanarity threshold fixed at 10°, we can embed a message of 575 bits maximum. After the insertion we compute the Hausdorff distortion in order to have information on the quality of the insertion. In this case, the Hausdorff distortion equals to $3.94 \cdot 10^{-4}$ for 216 bits embedded. For a better distortion, we can fixed the threshold at 5°, we can embed the same message for a Hausdorff distortion equals to $1.08 \cdot 10^{-4}$, but we can embed 264 bits maximum.

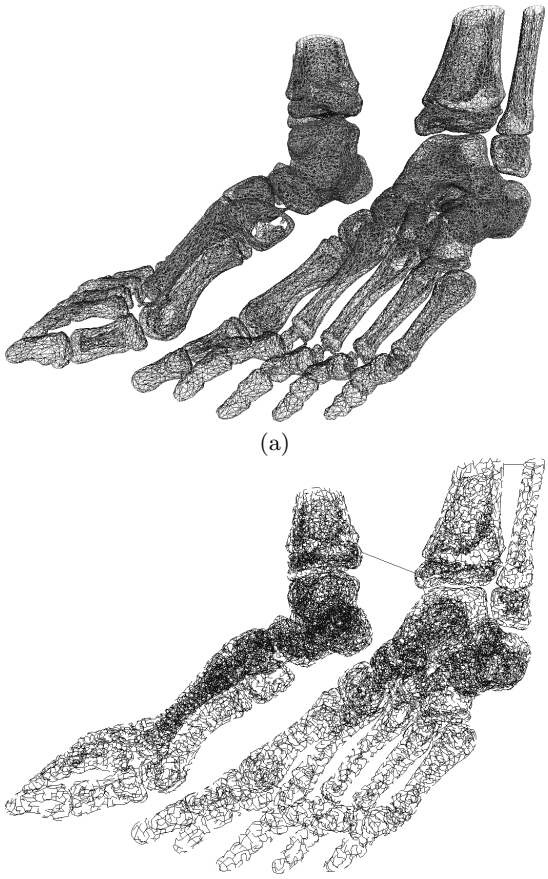
3.3 Perspectives

The method is interesting because we keep the watermark mesh as closed as possible from the original by modifying the connection only. Nevertheless, the complexity is very high (quadratic as a function of the number of vertices).

It is very fragile to noise addition, so we focus on this method to improve it by studying the possible displacement of the vertex in order to be more robust. As a function of the Prim's algorithm and the seed vertex, we are able to know how can a vertex be moved without editing the connections in the EMST [13]. Indeed, we watermark our message in the most robust areas.

4 Conclusion

We have presented a watermarking overview for medical application, in order to protect the 3D images or to enhance the 3D object by embedding metadata. In medical application the data are large for each patient, so we need to develop fast algorithms without increasing the size of the data.



(a) (b)

Fig. 10. An example of EMST applied on a simplified 3D mesh (48′711 vertices and 98′614 faces).

We have introduced 2D watermarking techniques used for 3D images. It is a low complexity algorithm with high complexity to enhance the 3D model. But it is very fragile to noise addition and it is impossible to extract the message from the 3D mesh.

Then, we have presented a 3D watermark method based on a mesh. The originality is to watermark a message without moving any vertices and without increasing the file size. The synchronization is made by the localization of quadruples in the EMST of the vertex cloud. The method is also fragile with low capacity of insertion, but the 3D mesh is watermarked.

We study the EMST structure in order to find new robust criterion in order to select robust areas and improve the computational time.

References

1. P. Amat, W. Puech, S. Druon, and J.P. Pedeboy. Lossless 3D Steganography Based on MST and Connectivity Modification. *Signal processing. Image communication*, 25(6):400–412, 2010.
2. C.K. Chan and LM Cheng. Hiding Data in Images by Simple LSB Substitution. *Pattern Recognition*, 37(3):469–474, 2004.
3. S. Cheng, Q. Wu, and K.R. Castleman. Non-Ubiquitous Digital Watermarking for Record Indexing and Integrity Protection of Medical Images. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2. IEEE, 2005.
4. P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174. Wiley Online Library, 1998.
5. G. Coatrieux, L. Lecornu, B. Sankur, and C. Roux. A Review of Image Watermarking Applications in Healthcare. In *Proc. of the 28th International Conference IEEE Engineering in Medicine and Biology Society*, 2006.
6. A. Kerckhoffs. La Cryptographie Militaire. *Journal des sciences militaires*, 9(5-38):161–191, 1883.
7. D. Koller, B. Frischer, and G. Humphreys. Research Challenges for Digital Archives of 3D Cultural Heritage Models. *Journal on Computing and Cultural Heritage (JOCCH)*, 2(3):1–17, 2009.
8. W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, page 169. ACM, 1987.
9. X. Mao, M. Shiba, and A. Imamiya. Watermarking 3D Geometric Models Through Triangle Subdivision. In *Proc. SPIE*, volume 4314, pages 253–260, 2001.
10. R. Ohbuchi, H. Masuda, and M. Aono. Watermaking Three-Dimensional Polygonal Models. In *Proceedings of the Fifth ACM International Conference on Multimedia*, pages 261–272. ACM New York, NY, USA, 1997.
11. B. Planitz and A. Maeder. Medical Image Watermarking: A Study on Image Degradation. In *Proceedings of the Workshop on Digital Image Computing: Technqiues and Applications*, pages 3–8, 2005.
12. W. Puech and JM. Rodrigues. A New Crypto-Watermarking Method for Medical Images Safe Transfer. In *Proc. of the 12th European Signal Processing Conference*, pages 1481–1484, 2004.
13. N. Tournier, W. Puech, G. Subsol, and J-P. Pedeboy. Sensitivity Analysis of Euclidean Minimum Spanning Tree. In *SPIE'10: 3D Image Processing (3DIP) and Applications, San Jose, USA (CA)*, 2010.
14. A. Wakatani. Digital Watermarking for ROI Medical Images by Using Compressed Signature Image. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2043–2048. IEEE, 2002.

Feature Vertices for 3D Synchronization Using Euclidean Minimum Spanning Tree

N. Tournier[a, b], W. Puech[a], G. Subsol[a] and J-P. Pedeboy[b]

[a] University of Montpellier II, LIRMM, UMR CNRS 5506,
161 rue Ada, 34095 Montpellier, France;

[b] Strategies S.A., 41-43 rue de Villeneuve,
Parc d'affaires SILIC - BP 80429, 94583 Rungis Cedex, France

ABSTRACT

Synchronization in 3D data hiding is one of the main problems. We need to know where we can embed information, and be able to find this space in order to extract the message. Various algorithms propose synchronization techniques by triangle or vertex path in a 3D mesh. In this paper, we proposed a new synchronization technique based on Euclidean minimum spanning tree computing (EMST) and the analysis of the displacement of the vertices without moving the connections in the tree. Based on the analysis of the vertices, we select the most robust vertices and synchronize these areas by computing a new EMST called "robust EMST". Then, we analyze the robustness of the technique, i.e. the stability of the most robust vertices selection; and demonstrate the consistence of the criterion selection with the vertex displacement.

Keywords: Feature Vertices, 3D Synchronization, 3D Watermarking, Graph Theory

1. INTRODUCTION

3D multimedia content is now everywhere, in the industry for modeling and design; in the entertainment for video gaming, cinema, in cultural heritage for 3D interaction, archiving, etc. Thus, large quantities of multimedia data are exchanged that raises some research challenges such as the protection of 3D objects. There are different ways to protect multimedia file. On the one hand, we can choose to encrypt the content; the communication will be secured until the customer decodes the information. On the other hand, we can embed a hidden message in the host signal (sound, image, video, 3D object).

One of the major problems of 3D data hiding is the synchronization. This step characterizes an insertion subspace for the message to hide. At the opposite of digital audio and image watermarking there is no manner to find a natural regular and unique path in a 3D object. In the spatial domain, there are various kinds of synchronization techniques. The idea is to find a unique path of vertices or facets in the mesh. In,[1] they propose to double a band of triangles depending on the message to embed. This method is not sure because it is very easy to find the message by identification of the double triangles. It is one of the first methods of data hiding for 3D meshes, with a lot of problems, which does not allow to embed long message. Other approaches[2–4] proposed to scan the mesh by a determinist sequence of triangles, independent of the message and of the insertion process. The synchronization stands on the choice of the initial triangle. This kind of approach depends on the connectivity of the mesh; therefore it is very fragile to topology modification such as remeshing, filtering, etc.

In this paper, we focus on the 3D synchronization and propose an improvement of a previous technique stands on the Euclidean minimum spanning tree (EMST) computing.[5] They define a structure that depends only on the vertices. The synchronization stands on the choice of one vertex only and does not move any vertex. It is robust to remeshing, but not in vertex decimation and very fragile to perturbation of the vertices such noise addition for example.

In Section 2, we present our synchronization technique. Section 4, we compare our experimental results and present a discussion in Section 3.

Further author information: (Send correspondence to N.T., W.P, or G.S)
E-mail: {nicolas.tournier, willam.puech, gerard.subsol}@lirmm.fr

2. FEATURE ROBUST VERTEX

We give prominence to the possible displacement of a vertex without changing the connexions in the Euclidean minimum spanning tree (EMST).[6] By this analysis, we need to know if these regions are robust, in the watermarking point of view, and how it is possible to synchronize them. We divide the problem in two parts. First, we need to characterize the most robust areas. So we need to find a criterion as a function of the displacement radius (r_i^+ and r_i^-) computed in.[6] Secondly, until the robust areas are identified, we need to synchronize them. In other words, we need to find a unique path between them. This path must also be robust to be able to find the correct message embedded.

We need to find a robust criterion to quantify the displacement of the vertices. Moreover we must have a good correlation with the perturbation of the vertices by the Gaussian noise addition with a standard deviation σ, in order to be able to find the most robust vertices again.

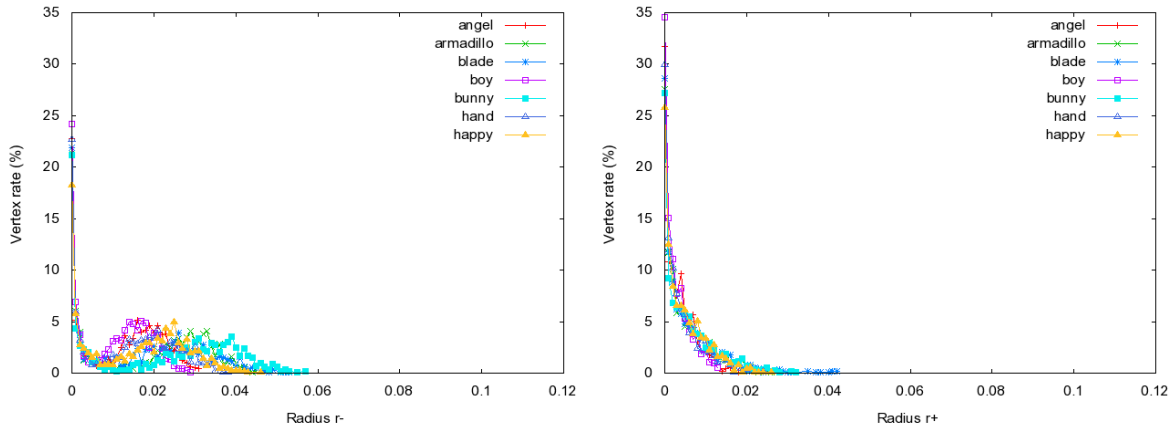


Figure 1. Distribution of the radius r^- and r^+. r^- stands for the distance which the vertex can be moved from its father and r^+ which the vertex can be moved away. More the distance is important, more the vertex is mobile.

From r^+ and r^-, illustrated Figure 1, it is easy to deduce a magnitude that quantifies the displacement of the vertices along their straight line. Let us denote by r this magnitude:

$$r_i = \min\{r_i^+, r_i^-\}.$$

Nevertheless, in[6] we consider only one degree freedom displacement, along a straight line. So we need to verify if this criterion can be an approximation of the 3D vertex displacement. From r and σ, we need to find a criterion correlated to the Gaussian noise. We denote by $r_{x\%}$ the value such as $x\%$ of the vertices verify $r_i > r_{x\%}$. That represents a threshold to the most robust vertices. This value will be an estimation of the robustness of the vertices.

The question is to know if the more robust vertices in the EMST are robust in the synchronization point of view. In other words, we want to be able to find the same vertices or the same areas after the perturbation by a Gaussian noise addition.

With this study, we are able to locate the most mobile vertices, and to synchronize these areas by find a path in the EMST of the robust vertices. The synchronization scheme is illustrated Figure 2. In the following Section, we present our experiment results that prove the improvement of synchronization techniques based on EMST.

3. EXPERIMENTAL RESULTS

We experiment our theory on one dozen of normalized 3D meshes composed by around thousands of vertices. We want to estimate the robustness of the synchronization process, i.e. the selection of the more robust vertices and in the same time the stability of the "robust EMST". So we proceed as follow:

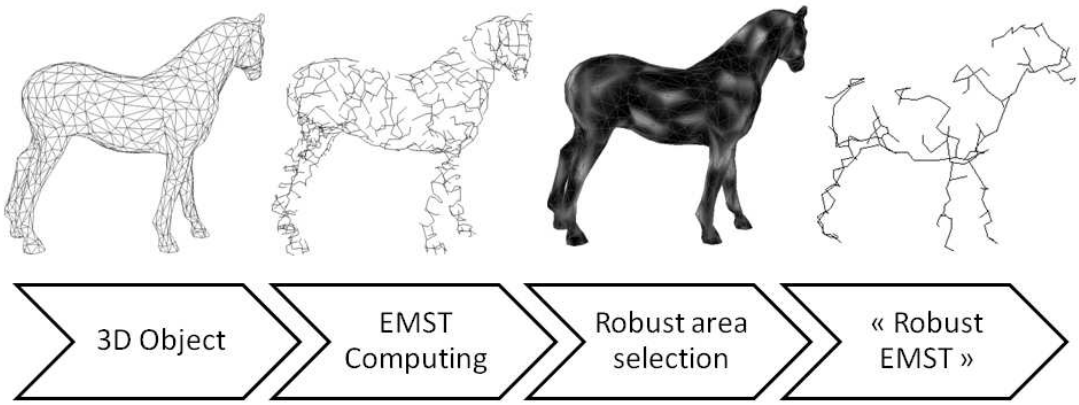


Figure 2. Representation of the synchronization process on *Horse*. We illustrate respectively from the left to the right: the 3D mesh, the EMST computed by Prim's algorithm,[7] the selection of the most mobile vertices represented in light colors, the synchronization of the most robust area by an other EMST computing. We called this tree, the "robust EMST".

1. From the original mesh M, we disturb the vertices by an additive Gaussian noise ($\mathcal{N}(\sigma, 0)$). We denote by M_σ, the noised mesh;
2. We compute the EMST T of the mesh M and respectively T_σ for M_σ;
3. We estimate for each vertex the possible displacement r_i without changing the EMST;
4. We select $x\%$ of the most robust vertices such as $r_i > r_{x\%}$ and we denote by $V_{x\%}$ and $V_{\sigma,x\%}$ the set of the selected vertices;
5. We compute EMST $T_{x\%}$ (respectively $T_{\sigma,x\%}$) on the set $V_{x\%}$ (respectively $V_{\sigma,x\%}$), we will call these EMST the "robust EMST";
6. We compare the two EMST;
7. For fixed σ we iterate N times since the step 2.

Here, the difficulty is to compare the two "robust EMST". The idea is to match each edge of T_σ to $T_{\sigma,x\%}$ such as we are able to compare the connections in the trees. If this function is a bijection between T_σ and $T_{\sigma,x\%}$, we compute the number of common edges between T_σ and $T_{\sigma,x\%}$. We denote by $\mu(T_\sigma, T_{\sigma,x\%})$ this measure. If T_σ and $T_{\sigma,x\%}$ are not in bijection, there is no compatibility so we return $\mu(T_\sigma, T_{\sigma,x\%}) = 0\%$. This is a simple heuristic to estimate the good correspondence between the two trees. It may be performed, but it is enough to prove our more robust synchronization.

We illustrate Figure 3, the results of the common edge rate ($\mu(T_{x\%}, T_{\sigma,x\%})$) of the "robust EMST" in average for (20 iterations) as a function of Gaussian noise intensity (σ).

As we can see, the "robust EMST" is quite robust. For the majority of the tested objects we preserve a good synchronization level until a Gaussian noise with a standard deviation σ around 10^{-5} or 10^{-4}. This is really interesting, at the opposite for a synchronization based only on the EMST[5] in which we are desynchronized for small intensity noise (around 10^{-7}, 10^{-6}) for the same objects.

The robustness can be explain by the corelation between $r_{x\%}$ and the intensity of the Gaussian noise. Let $\sigma_{x\%}$ is the critical standard deviation of the Gaussian noise such as $\mu(T, T_\sigma) = x\%$ in average. By the experience, we take one dozen of 3D objects composed of thousands vertices and fixe x to compute for each mesh $r_{x\%}$ and $\sigma_{x\%}$ for around 20 experimentations.

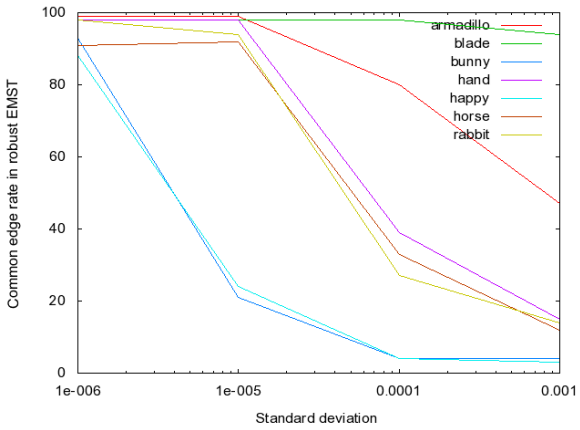


Figure 3. Representation of the common edge rate in robust EMST $\mu(T_{x\%}, T_{\sigma,x\%})$ as a function of the standard deviation of the Gaussian noise σ realized for 20 experiments.

According to Figure 4, we represent, for different selection rate x, the critical standard deviation as a function of our theoretical criterion. For each selection rate we obtain a straight line. Indeed, it exists a linear relation between $r_{x\%}$ and $\sigma_{x\%}$ that does not depend on x:

$$\sigma_{x\%} = k \cdot r_{x\%}.$$

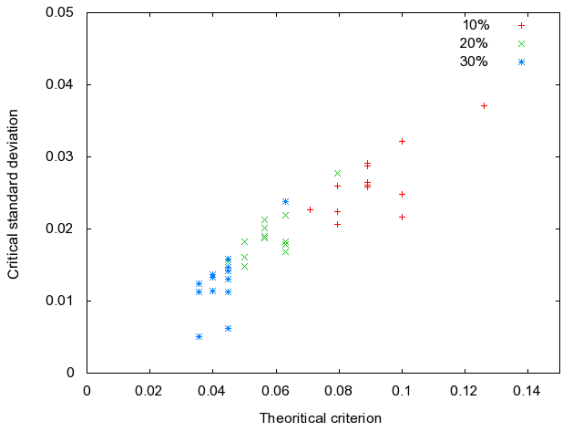


Figure 4. Correlation between our theoretical criterion $r_{x\%}$ and $\sigma_{x\%}$ the critical standard deviation.

By linear regression we estimate k around 3 for $x \in \{10; 20; 30\}$. In future work, it will be interesting to formalize this study in theoretical context. This kind of techniques should be interesting with various criterions that we are studying, such as the estimation of the discrete curvature. The aim is to find a stable criterion in order to use the synchronization in a robust watermarking scheme.

4. CONCLUSION

We propose an improvement of the synchronization by EMST.[5] Thanks to the analysis of the displacement of the vertices without changing the connections in the tree,[6] we propose a theoretical criterion in consistence with the Gaussian noised intensity in order to select robust areas. To synchronize them, we compute a new EMST based only on the most robust vertices. This new EMST is more robust. From the previous method, the synchronization process is more robust. But if we used the same embedding techniques in the most robust areas only, the capacity is potentially 3/10 less important for the same imperceptibility.

In the future, we propose to find different criterions stable to synchronization attacks based on the curvature, for example, in different domains such as the 3D wavelets.

REFERENCES

[1] Ohbuchi, R., Masuda, H., and Aono, M., "Watermaking Three-Dimensional Polygonal Models," in [*Proceedings of the Fifth ACM International Conference on Multimedia*], 261–272, ACM New York, NY, USA (1997).

[2] Cayre, F. and Macq, B., "Spatial Watermarking of 3D Triangle Meshes," in [*Proceedings of SPIE*], **4472**, 155 (2001).

[3] Mao, X., Shiba, M., and Imamiya, A., "Watermarking 3D Geometric Models Through Triangle Subdivision," in [*Proc. SPIE*], **4314**, 253–260 (2001).

[4] Rossignac, J., "Edgebreaker: Connectivity Compression for Triangle Meshes," *IEEE transactions on visualization and computer graphics* **5**(1), 47–61 (1999).

[5] Amat, P., Puech, W., Druon, S., and Pedeboy, J.-P., "Lossless 3D Steganography Based on MST and Connectivity Modification," *Signal Processing: Image Communication* **25**, 400–412 (July 2010).

[6] Tournier, N., Puech, W., Subsol, G., and Pedeboy, J., "Sensitivity Analysis of Euclidean Minimum Spanning Tree," in [*Proceedings of SPIE*], **7526**, 75260G (2010).

[7] Prim, R., "Shortest Connection Networks and Some Generalizations," *Bell System Technical Journal* **36**, 1389–1401 (1957).

DOCTISS 2011

"Filigrane" numérique pour les maillages 3D

Nicolas TOURNIER [1] (Doctorant 3eme année, informatique)
William PUECH [2], Gérard SUBSOL [3], Jean-Pierre PEDEBOY [4] (Directeurs de thèse)

Laboratoire d'Informatique de Robotique et de Micro électronique de Montpellier
Équipe ICAR (Image et Interaction)
Université de Montpellier II
34095 MONTPELLIER cedex, France

[1]nicolas.tournier@lirmm.fr
[2]william.puech@lirmm.fr, [3]gerard.subsol@lirmm.fr, [4]jp.pedeboy@cadwin.com

Mots clés — maillages 3D, insertion de données cachées, stéganographie, tatouage

Depuis plusieurs années, dans le domaine de la conservation du patrimoine, de nombreuses quantités de données sont échangées. Dans le but d'enrichir les données multimédia sans augmenter son espace mémoire, l'insertion de données cachées (ou tatouage numérique) peut être une solution. Du fait que les données sont très volumineuses, nous devons utiliser des algorithmes de faible complexité.

Le tatouage numérique a d'abord été appliqué sur des fichiers audio, puis sur des images et des vidéos. Depuis environ dix ans, quelques méthodes ont été proposées pour la 3D.

Dans cet article, nous présentons l'intérêt du tatouage en section I.. Puis nous proposons deux types de méthodes de dissimulation 3D. La première se base sur des acquisitions volumiques (CT-scan), dans chaque coupe nous insérons de l'information par des méthodes classiques issues du tatouage 2D. L'autre méthode de dissimulation est faite sur des acquisitions surfaciques. Dans ce cas nous nous servons des connexions du maillage pour insérer de l'information. Ces deux méthodes seront détaillées dans les sections II. et III. accompagnées de quelques résultats expérimentaux.

I. RÉSUMÉ DÉTAILLÉ

De nos jours, la création de modèles 3D volumineux pour la conservation du patrimoine ne cesse d'augmenter ce qui soulève des problématiques intéressantes. Selon [1], les principaux challenges à relever pour les modèles graphiques 3D sont :

- la gestion des droits numériques ;
- la gestion des méta données ;
- l'indexation et la recherche dans les bases de données.

L'insertion de données cachées peut être un bon compromis pour répondre à ces problématiques.

I.1. Vulgarisation du tatouage numérique

Prenons un billet de banque, tout le monde sait que lorsqu'on le regarde à la lumière le filigrane apparaît. On peut faire correspondre cette métaphore avec le tatouage numérique, où le billet de banque est une image, ou un objet 3D par exemple ; qu'on appelle le signal hôte. Le filigrane est le message caché dans le signal hôte.

En poursuivant dans cette analogie, pour ajouter le filigrane le billet a dû être imprimé d'une manière bien particulière avec du matériel adapté. Le choix du matériel et la disposition du filigrane correspondent au processus de synchronisation. C'est l'étape qui identifie les zones dans laquelle on peut insérer de l'information. Alors l'impression à proprement parlé est l'algorithme d'insertion du message. La figure 1 illustre le processus complet d'insertion du message.

Pour voir le filigrane, il suffit de regarder le billet à la lumière. Cette action est par analogie l'algorithme d'extraction du message. La figure 2 illustre le processus complet d'extraction du message.

I.2. Applications

En fonction du rôle attribué au message, on obtient des applications variées.

Dans le but de garantir l'authenticité d'un document ou dans la gestions des droits d'auteur par exemple, le message peut être une signature numérique. L'information cachée doit pouvoir être récupérée même si le signal hôte a subi des transformations. Nous parlons de tatouage robuste et dans le cas de données géométriques, ces déformations peuvent être des translations, des rotations, des homothéties, un lissage, une compression, etc.

On peut souhaiter identifier l'acheteur d'un contenu légal en dissimulant des empreintes numériques qui identifient de manière unique l'acheteur. En cas de distribution illégale, le responsable peut être retrouvé. Bien que ce genre d'application ressemble à la précédente, les problématiques sont différentes. Nous appelons des applications de recherche de traitres (ou *fingerprinting*), où nous pouvons imaginer un groupe d'utilisateurs malveillants (les traitres) unir leurs copies tatouées afin d'en produire une nouvelle ressemblant aux originales et privée de marquage.

De plus, il est possible d'enrichir la donnée en insérant ses méta données telles que des commentaires ou des données bibliographiques par exemple. Contrairement aux applications précédentes, nous n'avons pas besoin de méthodes robustes, mais de pouvoir insérer beaucoup d'informations sans augmenter la taille du signal hôte, nous parlons d'algorithme à haute capacité. Il est alors possible dans un maillage 3D, de dissimuler des textures, des sons, un objet de basse résolution par exemple.

Comme nous le montrons, les possibilités d'utilisation du tatouage numérique sont multiples. Néanmoins, pour chaque application il faut trouver un compromis entre la capacité (le

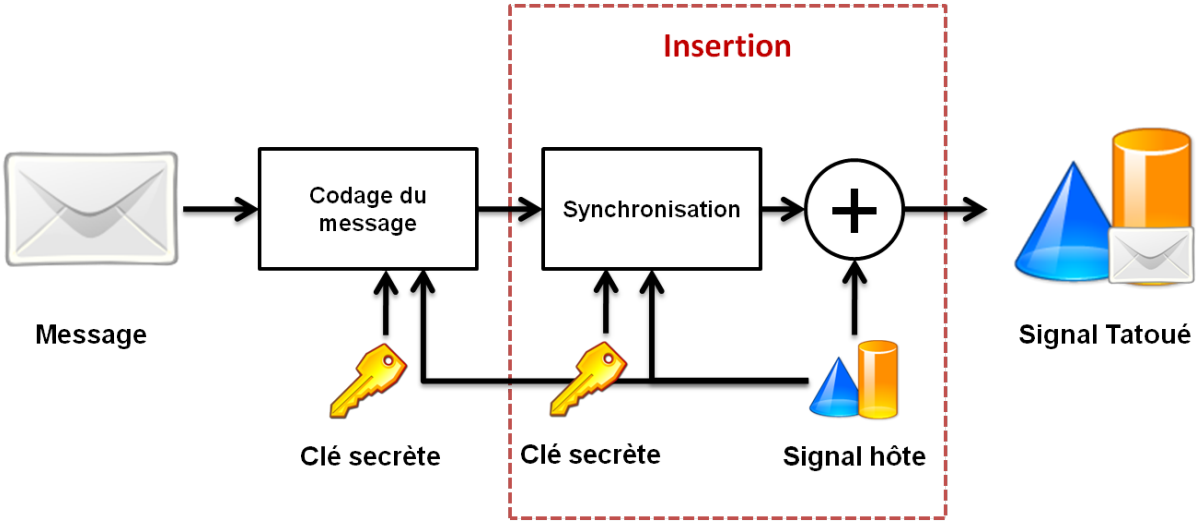


FIGURE 1 : Algorithme d'insertion de l'information.

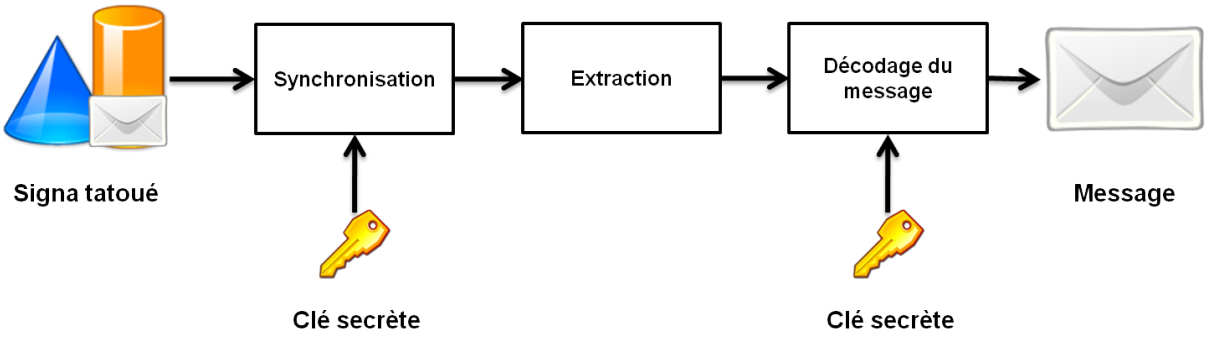


FIGURE 2 : Algorithme d'extraction de l'information.

nombre maximum de bits qu'il est possible de dissimuler), l'imperceptibilité, la robustesse, la sécurité de l'algorithme (au sens de Kerckhoffs [2]) et sa complexité algorithmique. Les données étant généralement très volumineuses, il faut des algorithmes de faible complexité.

Nous avons développé diverses méthodes de tatouage 3D. Nous présentons en section II. une méthode se basant sur des données volumiques qui correspondent à une séquence d'images 2D et en section III. une méthode qui se base sur un maillage 3D surfacique.

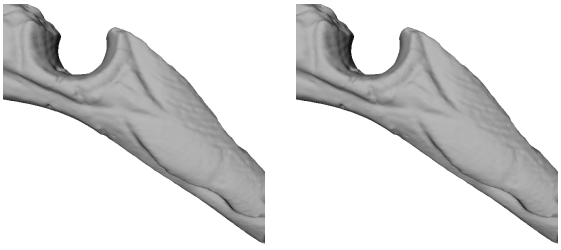
II. INSERTION DE DONNÉES CACHÉES SUR DES IMAGES 3D

Nous travaillons des images 3D où chaque coupe est équivalente à une image 2D. Pour cette approche, l'idée est d'utiliser des algorithmes bien connus du tatouage 2D sur chaque couche.

Nous avons choisi une méthode simple pour tatouer les images 2D. Il suffit de substituer le dernier bit significatif (LSB) de chaque pixel par un bit du message que l'on veut dissimuler [3]. A l'extraction, il suffira de lire les pixels dans le bon ordre et de récupérer le LSB pour reconstruire le message.

Une fois les images tatouées, nous construisons un modèle surfacique équivalent [4]. La figure 3 illustre la reconstruction d'un maillage à partir d'une image 3D originale et de l'image 3D tatouée. On ne remarque aucune différence perceptuelle entre les deux maillages.

Expérimentalement, nous avons une image 3D du *"bâton percé"* composée de 125 coupes de dimension (382 x 82 pixels) dans laquelle nous insérons une image 2D de 25.8ko. La figure 4.a montre la coupe originale n^{o}54, 4.b la même coupe tatouée et 4.c est la différence entre les deux images. Nous mesurons la distorsion entre les deux images en calculant le PSNR [1]. En notant i_k et j_k les valeurs du k-ième pixel dans les images I et J composées chacune de N pixels , on calcule l'erreur quadratique moyenne $\mu(I,J)$:



(a) (b)

FIGURE 3 : Le maillage 3D reconstruit à partir de l'image 3D originale (a) et de l'image 3D tatouée (b).

$$\mu(I,J) = \frac{1}{N}\sum_{k=1}^{N}(i_k - j_k)^2,$$

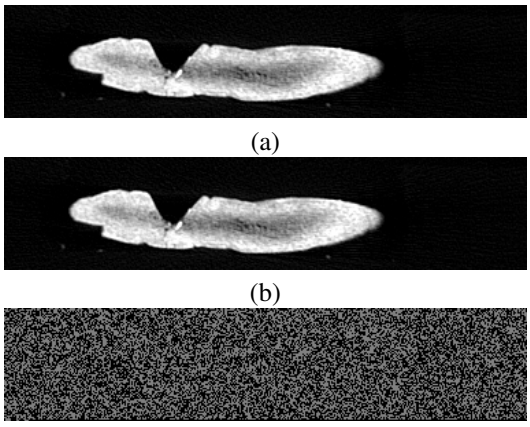
et on obtient le PSNR, exprimé en décibel (dB), par :

$$PSNR(I,J) = 10 \cdot \log(\frac{255}{\mu(I,J)}).$$

Le PSNR mesure la dégradation de l'image, ainsi pour une valeur supérieure à 40 dB on considère que la modification est imperceptible.

Lors de notre expérience, nous obtenons en moyenne un PSNR de 63.8 dB. En prenant l'image originale et l'image tatouée, il est impossible de savoir dans laquelle se trouve une information cachée.

1. Peak Signal to Noise Ratio



(a) (b) (c)

FIGURE 4 : Comparaison de la coupe n^o 54 du "bâton percé" originale (a), de la même coupe tatouée (b) et l'image différence (c). PNSR = 51.2 dB.

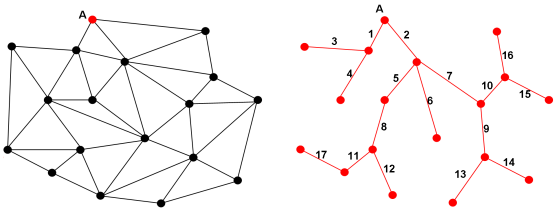
De la même façon, on compare les maillages reconstruits (figure 3). Il n'y a strictement aucune différence entre les deux modèles. Cependant il est impossible de récupérer le message dissimulé à partir du maillage, on ne peut l'extraire uniquement à partir de l'image 3D.

Pour conclure sur cette méthode, c'est un algorithme très rapide et très simple de tatouage d'image 3D qui peut être utilisé pour des applications d'enrichissement de données.

III. INSERTION DE DONNÉES CACHÉES SUR DES MAILLAGES 3D

Nous avons montré précédemment comment tatouer un objet 3D volumique en s'inspirant des techniques du tatouage 2D. Il serait intéressant de modifier directement le maillage pour insérer de l'information. De nombreuses techniques existent, la plupart d'entre elles étant robustes aux transformations géométriques rigides.

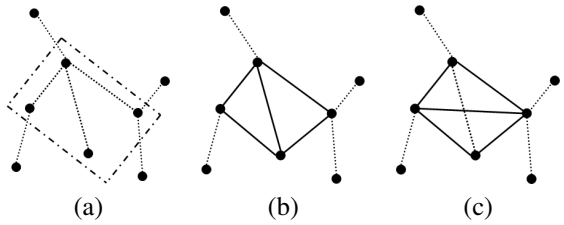
Peu de méthodes ne déplacent aucun point du maillage [5, 6, 7]. L'originalité de [5] est de modifier uniquement les connexions entre les sommets, sans augmenter la taille du fichier et sans déplacer un seul point. Pour synchroniser, c'est-à-dire trouver un ordre sur l'ensemble des points, ils utilisent un arbre couvrant de poids minimum [8] (Euclidean Minimum Spanning Tree, EMST). C'est une structure qui pour un ensemble de points est unique, et à partir d'un sommet racine le parcours des sommets est unique (fig. 5). Ce qui rend cet objet mathématique très intéressant pour la synchronisation [9, 10].



(a) (b)

FIGURE 5 : Un exemple de maillage (a) et de son arbre couvrant de poids minimum associé (b). On remarque que les connexions dans l'arbre ne correspondent pas nécessairement aux arêtes du maillage.

Dans l'arbre, nous recherchons des quadruples (sommets de degré 4) dans lesquels nous dissimulons un bit d'information. L'insertion est assez simple, si nous voulons dissimuler un 0 l'arête qui se trouve dans l'arbre se trouve aussi dans le maillage. Sinon, pour insérer un 1 on choisit l'autre arête. Ce processus d'insertion est illustré sur la figure 6.



(a) (b) (c)

FIGURE 6 : Illustration de l'arbre couvrant de poids minimum (a), du maillage lors de l'insertion d'un bit à 0 dans un quadrangle sélectionné (b) et d'un bit à 1 (c).

Afin de ne pas créer d'importantes distorsions perceptuelles et de problème de synchronisation, les quadruples doivent vérifier ces conditions :

- Coplanarité : les deux triangles doivent être planaires (fig. 7), un seuil de sélection des quadrangles est fixé ;
- Convexité : le quadruple doit être convexe car en cas de permutation d'une arête la surface recouverte doit être proche de l'originale (fig. 8) ;
- Couverture : si deux quadruples sont voisins, alors un seul est choisi pour le tatouage.

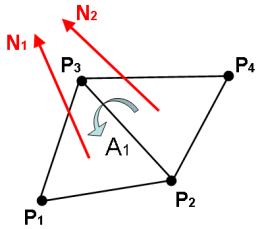
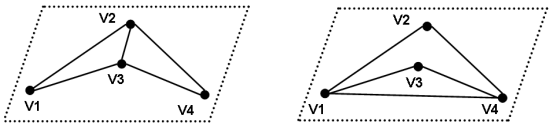


FIGURE 7 : Illustration de la contrainte de coplanarité.



(a) (b)

FIGURE 8 : Illustration de la contrainte de convexité.

De la même manière qu'on mesure une dégradation de l'image avec le PSNR, dans le cas de la 3D on utilise la distance de Hausdorff. En prenant deux surfaces similaires, cette valeur correspond à la plus grande distance qui les sépare l'une de l'autre.

Cependant, au lieu d'évaluer la distorsion sur la position des points (qui serait nulle car aucun point n'est déplacé), nous évaluons l'imperceptibilité sur la distorsion des centres de gravité de chaque facette.

Le nombre de quadrangles sélectionnés dépend du seuil de coplanarité fixé initialement. Plus le seuil est bas, plus on force les quadrangles à se trouver dans le même plan et ainsi la capacité d'insertion diminue. Pour un seuil fixé à 1°, on peut insérer au maximum 4466 bits dans le *"bâton percé"* (composé de 98251 sommets et 197214 facettes) ce qui entraîne une distorsion maximale de 0.001637. En d'autres mots, pour une diagonale de 10 cm, l'erreur maximale est de 1 mm. Respectivement pour un seuil fixé à 30°, on peut insérer 21698 bits avec une distorsion maximale de 0.029063.

La méthode est intéressante car on garde le maillage aussi fidèle que possible à l'original en ne modifiant que les connexions. Cependant, la complexité de l'algorithme est trop importante (quadratique en fonction du nombre de sommets).

De plus la méthode est très sensible aux modifications. Nous nous sommes intéressés par la suite à l'amélioration de

cette technique par l'étude du déplacement possible des points tout en conservant le même arbre [9].

IV. Conclusion

Le tatouage peut être utilisé dans le cadre d'applications pour la conservation du patrimoine afin de protéger et enrichir les modèles 3D. Les données sont volumineuses, donc nous devons développer des algorithmes rapides.

Nous avons montré deux approches différentes pour la dissimulation d'information dans les données 3D qu'elles soient volumiques en s'inspirant des techniques bien connues du tatouage 2D ou surfaciques sans modifier la position des points dans le maillage.

Références

[1] D. Koller, B. Frischer, and G. Humphreys. Research Challenges for Digital Archives of 3D Cultural Heritage Models. *Journal on Computing and Cultural Heritage (JOCCH)*, 2(3) :1–17, 2009.

[2] A. Kerckhoffs. La Cryptographie Militaire. *Journal des sciences militaires*, 9(5-38) :161–191, 1883.

[3] C.K. Chan and LM Cheng. Hiding Data in Images by Simple LSB Substitution. *Pattern Recognition*, 37(3) :469–474, 2004.

[4] W.E. Lorensen and H.E. Cline. Marching Cubes : A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, page 169. ACM, 1987.

[5] P. Amat, W. Puech, S. Druon, and J.P. Pedeboy. Lossless Data Hiding Method Based on MST and Topology Changes of 3D Triangular Mesh. In *EUSIPCO'08 : 16th European Signal Processing Conference, Lausanne, Switzerland*, 2008.

[6] R. Ohbuchi, H. Masuda, and M. Aono. Watermaking Three-Dimensional Polygonal Models. In *Proceedings of the Fifth ACM International Conference on Multimedia*, pages 261–272. ACM New York, NY, USA, 1997.

[7] X. Mao, M. Shiba, and A. Imamiya. Watermarking 3D Geometric Models Through Triangle Subdivision. In *Proc. SPIE*, volume 4314, pages 253–260, 2001.

[8] R.C. Prim. Shortest Connection Networks and Some Generalizations. *Bell System Technical Journal*, 36 :1389–1401, 1957.

[9] N. Tournier, W. Puech, G. Subsol, and J-P. Pedeboy. Sensitivity Analysis of Euclidean Minimum Spanning Tree. In *SPIE'10 : 3D Image Processing (3DIP) and Applications, San Jose, USA (CA)*, 2010.

[10] N. Tournier, W. Puech, G. Subsol, and J-P. Pedeboy. Feature Vertices for 3D Synchronization Using Euclidean Minimum Spanning Tree. In *SPIE'11 : 3D Image Processing (3DIP) and Applications, San Francisco, USA (CA)*, 2011.

3D Multi Resolutions Synchronization Scheme Based on Feature Point Selection

N. Tournier[a, b], W. Puech[a], G. Subsol[a] and J-P. Pedeboy[b]

[a] UM2 - LIRMM - CNRS, Address, Montpellier, France;
[b] STRATEGIES S.A., Address, Rungis, France.

ABSTRACT

Multimedia protection is one of the main research challenges in computer sciences.[1] We can encrypt the media in order to make the content unreadable without a secret key of decryption, protect the file with Digital Right Management (DRM), or embed an hidden message in the file (watermarking and steganography). We are interested in data hiding applications for 3D mesh. In this domain, there a mainly problem: the synchronization. It is the operation that permits to scan a mesh with a unique path and by selecting the same areas (vertices, triangles, quadrangles, for example) before and after the embedding.

In this paper, we propose a new synchronization technique based on feature point selection in a low resolution of the object. The building of the low resolution is made by decimation[2] and the feature point selection is based on the discrete curvature computing. We evaluate the robustness of the synchronization in the low resolution and in the heigh resolution.

Keywords: 3D, watermarking, synchronization, feature point, multi resolution

1. INTRODUCTION

We are daily in touch with 3D; at the cinema, at television (cartoons, advertising), in video games, in the industry (CAD), etc. More and more 3D object are produce and exchange. We need to find protection for these files too. AS we know, there is differents manner to protect files. We can make the content unreadable with cryptographic application, until the customer take the secret key to decrypt the file. It is also possible to limit the using of the file with Digital Right Management (DRM). And, an other solution is to hide a message in the file; this is steganography and watermarking applications.

The major problem of data hiding for 3D content is the synchronization step. Synchronization permits to select some areas such as a list of vertices or a list of facets and order them in a unique manner to be able to find the same sets after the insertion of various allowed modification of the mesh.

In the spatial domain, most of the time we want to make the analogy with 2D images. By creating a bang of triangles,[3,4] it is possible to scan the mesh as we scan an image by lines or columns. With the same idea, inspired by Eadgebreaker[5] a lossless compression algorithm, high capacity method[6] was proposed. Other kind of synchronization are possible.

One of the most original[7] propose to do not move any vertices. The synchronization is made by computing a Euclidean Minimum Spanning Tree (EMST). This structure is very interesting because giving a set of vertices, the EMST is unique. Moreover, with a space orientation and a stating vertex, the scan of the tree is also unique. These uniqueness properties are necessary for synchronization applications.

Inspired by,[8] we create a new synchronization scheme that combines the idea of feature point selection and multi resolution aspect with EMST structure described in the next section.

Further author information: (Send correspondence to N.T, G.S, W.P)
N.T., G.S., W.P.: E-mail: nicolas.tournier@lirmm.fr, gerard.subsol@lirmm.fr, william.puech@lirmm.fr, Telephone: +33 (0)4 67 00 00 00
J-P.P.: E-mail: bba@cadwin.com, Telephone: +33 (0)1 41 73 04 80

2. PROPOSED METHOD

Let a set of meshes $\{M_{HR}, M_1, M_2, ..., M_{LR}, ..., M_l\}$ with M_{HR} the original mesh and the other are computed by multi resolution analysis. Multi resolution decomposition can be: wavelet decomposition, vertex decimation, or other techniques used in Level Of Detail (LOD) applications for example.

As for us, we choose vertex decimation, Quadric Edge Collapse Decimation[2] implemented in MeshLab[9] in order to have first results. From this decomposition, we create a simple parenthood relationship between the vertices in each mesh; such as the vertex $v_i^j \in M_j$ is a father of $v_k^{j-1} \in M_{j-1}$ if and only if v_k^{j-1} is in the Voronoi cell of v_i^j computed with the vertex of M_j.

Then, the selection of feature point is made in one of the mesh from the decomposition. Let j the chosen level and n_j the number of vertices in this resolution. We choose two selection criterion based on the curvature: the Gaussian curvature k_G and another criterion denote by $\tilde{k} = k_{min}^2 + k_{max}^2$; and we compare the results between these one. We fixed a selection rate $x \in [0; 1]$ and for each vertex v_i^j, we compute $\tilde{k}_i$ and $k_{G,i}$. Then we select the $x \cdot n_k$ vertices with the maximum curvature value, we denote by $V_{j,x}$ the set of the selected vertices.

After the selection, we need to synchronize the vertices. We choose to compute a Euclidean minimum spanning tree (EMST) based on $V_(j, x)$ with the Euclidean distance. The advantage of the EMST is unique for a set of vertices. Moreover, when we let a root vertex, it is possible to define a unique path in the tree. These are interesting properties for the synchronization. We prove theoretically and by the experience that EMST are fragile.[10] But if we select a set of vertices with a robust criterion, the robustness of the EMST is significantly better.[11] At this step of the synchronization technique, we have a set of selected vertices synchronize by an EMST in M_j. But the aim is to synchronize the higher resolution of the mesh, so we need to pass the EMST of the selected vertices on M_{HR}.

By the parenthood relationship, for each vertex in M_j corresponds to a set of vertices in M_{j-1}. We can extend the relationship by transitivity. Indeed, for each vertex in M_j, we associate a patch in M_{HR}. As the vertices are ordered, the patches are also.

In conclusion of the synchronized method description, this is a technique interesting for 3D insertion technique patch oriented. In the other case, our synchronized scheme can be extended. It is one of the first proposed feature point synchronization method using multi resolution with.[8]

3. EXPERIMENTAL RESULTS

We focus on the parameters of our method. Our input is this set of meshes $\{M_{HR}, M_1, M_2, ..., M_{LR}, ..., M_l\}$, in other words we assume the parameters of the multi resolution decomposition are fixed.

First, we can choose the level in which the selection is made. This first parameter is quite interesting to study because we need to find a compromise between the relevance of the curvature measure and the stability of the resolution.

The selection rate $x \in [0; 1]$ is a parameter that we can modify. We want to build a selection that verifies the condition: more x close to 0, more the selection is robust but there is few vertices selected. To be robust to cropping attack, it is not necessary to have a lot of vertices selected, or a lot of patches, they must be spread homogeneously in the mesh. Face to noise addition and filtering attacks for example, the robustness is more important than the quantity or the repartition of the vertices. In these conditions, we need to make another compromise.

We compare the influence of the selection rate. We select in the first experiment 10% of the vertices and 5% in the other. The experiment is built as follows:

- Gaussian noise addition $N(\sigma, \mu)$, with σ the standard deviation and μ the mean fixed to 0 to the original mesh M_{HR}, and we denote by $M_{HR,\sigma}$ the noised mesh;
- Building the set of multi resolution meshes $\{M_{HR,\sigma}, M_{1,\sigma}, M_{2,\sigma}, ..., M_{LR\sigma}, ..., M_{l,\sigma}\}$ with the same parameter than the original;

- Choosing the same resolution for the curvature computing; Selecting the robust vertices with the selection rate;
- Building the patches in the high resolution by the parenthood relationship;

By this experimentation, we take some 3D objects with around 10000 vertices, normalized in order to be including in a unitary box. We analyze the evolution 1, as a function of the standard deviation of the Gaussian noise varying between $[10^{-7}; 10^{-3}]$:

- false positive selection rate (i.e. counting the vertices that are selected in the noise mesh while they are not selected in the original),
- false negative selection rate (i.e. counting the vertices that are not selected in the noise mesh while they are selected in the original),
- and perfect matching.

These measures are made to compare the quality of the vertex selection in the low resolution mesh, and also in the high resolution to measure the quality of the vertex selected in each patch. For this first experimentation, we choose selection ratio equals to 5% and 10%; and the low resolution rate is obtained after one loop of Quadric Edge Collapse Decimation, reducing the number of vertices by 50%.

If we are looking to the perfect matching ratio, the results are interesting. For a noise with standard deviation $\sigma = 10^{-7}$, in the same order than the precision of the data, the synchronization is totally preserved (the ratio is very closed to 100%: $99,7\%$) in the low resolution mesh for selection rate equals to 5% or 10%.

For a visible noise addition ($\sigma = 10^{-5}$), we preserve only 80% of the vertices in LR and HR for a selection fixed at 10%. Moreover, if the ration is less important 5%, the result is worse. It is quite surprising, because the selection might be more robust, if and only if the criterion is robust.

To improve our method, we change the level of resolution in which the selection of the vertices is made. Moreover, in the selection of the vertices it will be interesting to choose its neighbors in the LR and then remove some vertex in the patch (it may be at the frontier) created in HR; because the selected error vertex is in general in the neighborhood of the original one and with this proposed improvement it is possible to limit the errors.

4. CONCLUSION

To conclude, we propose a new 3D synchronization method oriented feature point selection and multi resolution. It is one of the first method that combines these two approaches such as.[8] Feature points are selected in a low resolution of the object we want to watermark. The selection criterion is as function of the curvature of the vertices. The scheme of the method is quite interesting because we need to find compromises between the choice of the resolution, the criterion of selection and the ratio of the selected vertices.

This is only the synchronization process that is made to complete for a watermarking scheme. In order to complete the method, the embedding process has to preserve the selection criterion in the low resolution.

REFERENCES

[1] Koller, D., Frischer, B., and Humphreys, G., "Research challenges for digital archives of 3d cultural heritage models," *Journal on Computing and Cultural Heritage (JOCCH)* **2**(3), 1–17 (2009).

[2] Heckbert, P. and Garland, M., [*Survey of Polygonal Surface Simplification Algorithms*], Citeseer (1997).

[3] Mao, X., Shiba, M., and Imamiya, A., "Watermarking 3d geometric models through triangle subdivision," in [*Proc. SPIE*], **4314**, 253–260 (2001).

[4] Cayre, F., Rondao-Alface, P., Schmitt, F., Macq, B., and Maītre, H., "Application of spectral decomposition to compression and watermarking of 3d triangle mesh geometry," *Signal Processing: Image Communication* **18**(4), 309–319 (2003).

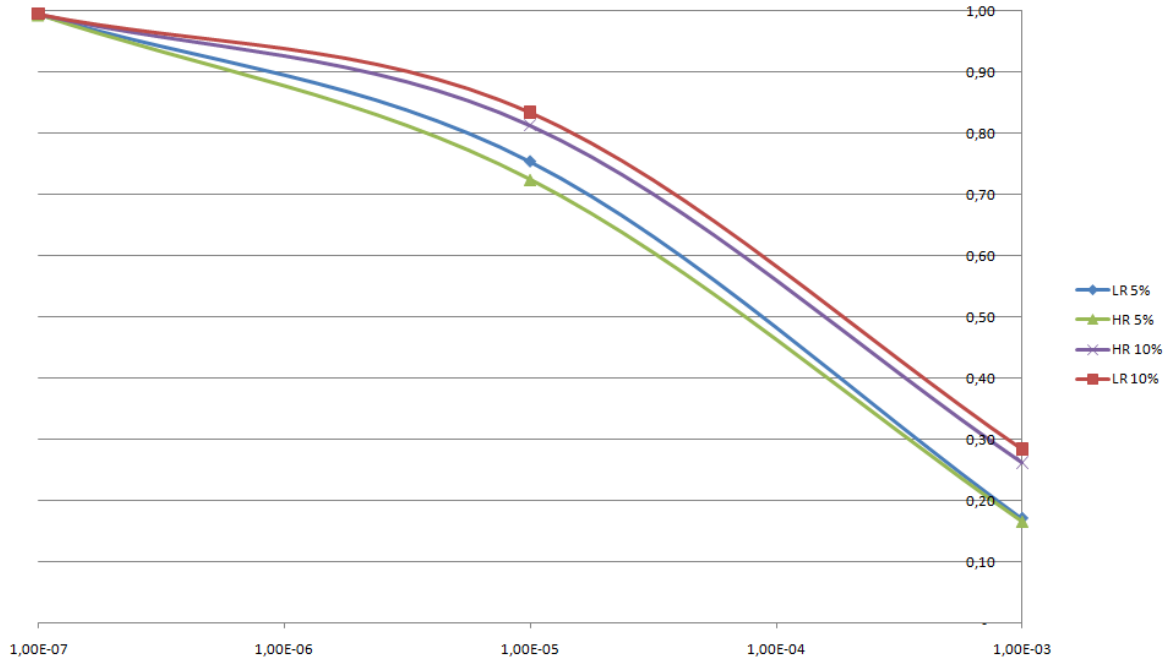


Figure 1. Representation of selection rate as a function of the standard deviation of the Gaussian noise.

[5] Rossignac, J., "Edgebreaker: Connectivity compression for triangle meshes," *IEEE transactions on visualization and computer graphics* **5**(1), 47–61 (1999).

[6] Bogomjakov, A., Gotsman, C., and Isenburg, M., "Distortion-free steganography for polygonal meshes," in [*Computer Graphics Forum*], **27**(2), 637–642, Citeseer (2008).

[7] Amat, P., Puech, W., Druon, S., and Pedeboy, J., "Lossless data hiding method based on mst and topology changes of 3d triangular mesh," in [*EUSIPCO'08: 16th European Signal Processing Conference, Lausanne, Switzerland*], (2008).

[8] Alface, P., Macq, B., and Cayre, F., "Blind and robust watermarking of 3d models: How to withstand the cropping attack?," in [*Image Processing, 2007. ICIP 2007. IEEE International Conference on*], **5**, V–465, IEEE (2007).

[9] Cignoni, P., Corsini, M., and Ranzuglia, G., "Meshlab: an open-source 3d mesh processing system," *ERCIM News* **73**, 45–46 (2008).

[10] Tournier, N., Puech, W., Subsol, G., and Pedeboy, J.-P., "Sensitivity analysis of euclidean minimum spanning tree," in [*SPIE'10: 3D Image Processing (3DIP) and Applications, San Jose, USA (CA)*], (2010).

[11] Tournier, N., Puech, W., Subsol, G., and Pedeboy, J.-P., "Feature vertices for 3d synchronization using euclidean minimum spanning tree," in [*SPIE'11: 3D Image Processing (3DIP) and Applications, San Francisco, USA (CA)*], (2011).

Abstract

Data security is one of the main issue in computer science. We need to develop solutions for confidentiality, communication, fingerprinting or identification applications for exemple. In this thesis made with STRATEGIES S.A., the chosen method to protect 3D meshes is watermarking. Watermarking is divided in two steps, the embedding and the extraction. In both of them a synchronization phase is needed. It is one of the most important step for 3D mesh because it permits to look for areas available to embed information, and order them. All the thesis is devoted to the synchronization step. First of all, we propose a classification of watermarking techniques based on the type of synchronization method instead of evaluation criterions such as robustness or capacity. Then, from methods based on Euclidean minimum spanning tree, we propose a theoritical analysis of the mobility of the vertices in that kind of structure. First, we explain the reasons of the sensibility of the structure. Secondly, we propose another scheme based on the Euclidean minimum spanning tree knowing its fragility.

Keywords: *3D mesh, watermarking, synchronization, graph theory, Euclidean minimum spanning tree.*

Résumé

De nos jours la protection des données numériques est un problème très important. Que ce soit pour des applications de confidentialité, de communication, de traçabilité ou d'identification par exemple, il est nécessaire de développer des techniques adaptées. Dans le cadre de cette thèse en collaboration avec la société STRATEGIES S.A., la méthode choisie pour la protection de maillages 3D est l'insertion de données cachées, également appelée tatouage numérique. Pour des données 3D, un des problèmes les plus importants est la phase de synchronisation qui intervient dans les algorithmes d'insertion et d'extraction des données. Cette phase permet de repérer, de sélectionner et d'ordonner les « zones » qui sont privilégiées pour la dissimulation d'information. Nous avons choisi d'orienter le manuscrit sur cette phase. Ainsi, nous proposons une classification des méthodes de tatouages en fonction de leur méthode de synchronisation. Puis en se basant sur des techniques de synchronisation par des structures de données, telle que les arbres couvrants de poids minimum, nous proposons une analyse théorique de cette structure. Dans un premier temps nous expliquons les raisons de la sensibilité des arbres à la mobilité des points. Puis connaissant ses faiblesses, nous proposons une autre technique de synchronisation toujours basée sur les arbres couvrants de poids minimum.

Mots clefs : *maillage surfacique, tatouage numérique, synchronisation, théorie des graphes, arbres couvrant de poids minimum.*

LIRMM — 161, rue Ada — 34095 Montpellier cedex 5 — France