CORESA 2013 - 16ème édition du colloque COmpression et REprésentation des Signaux Audiovisuels
Le Creusot, FRANCE, 28 et 29 novembre 2013

Construction d'un chemin Hamiltonien unique et robuste dans un nuage de points 3D.

V. Itier[1,3] W. Puech[1] G. Gesquière[2] J-P.Pedeboy[3]

[1] LIRMM UMR 5506 CNRS, Université Montpellier 2
161 rue Ada, 34095 Montpellier, FRANCE
{vincent.itier, william.puech}@lirmm.fr

[2] LIRIS, UMR CNRS 5205, Université Lumière Lyon 2
Bat C- 5 avenue Pierre Mendès France 69676 Bron FRANCE
gilles.gesquiere@liris.cnrs.fr

[3] STRATEGIES S.A
41-43 rue de Villeneuve, Parc des Affaires SILIC - BP 80429, 94583 Rungis FRANCE
jp.pedeboy@cadwin.com

Résumé

Une synchronisation efficace qui permettrait un ordonnancement des données, dans les modèles 3D maillés, est un défi majeur. Les modèles 3D sont de plus en plus utilisés via les technologies actuelles comme internet. Avec cette diffusion, la demande en compression, en visualisation et en protection augmente. Cependant, il n'existe toujours pas de méthode parfaite pour réaliser la synchronisation des données de façon aveugle et robuste. Nous proposons une méthode basée sur un chemin Hamiltonien pour synchroniser les données de nuages de points 3D. Nous montrons expérimentalement que cette méthode est rapide et robuste contre un bruit Gaussien. Cette nouvelle approche peut générer de meilleurs résultats que les méthodes s'appuyant sur le maillage car elle n'utilise pas la connectivité des sommets.

Mots clefs

Nuage de points 3D, Synchronisation, Ordonnancement, Robustesse.

1 Introduction

Les technologies actuelles de partage telles qu'Internet permettent l'accès, la visualisation, la modification, l'impression et le partage de données numériques en un seul clic. Les dernières avancées dans l'acquisition 3D, la modélisation géométrique et le rendu 3D, ont produit de nombreux modèles 3D, notamment dans des applications comme l'imagerie médicale, la conception assistée par ordinateur (CAO) et le divertissement numérique. Il est donc essentiel d'assurer la sécurité de ces modèles 3D pour maintenir leurs droits d'auteur. Afin de protéger les données numériques, la dissimulation de données est une méthode efficace dans laquelle un message est inséré dans les données du média hôte. L'insertion de données cachées doit permettre de retrouver ces données après des transformations ou des attaques. Dans certains cas, une méthode dite "fragile" est utilisée pour l'authentification du contenu [1]. Pour plus d'informations, les schémas d'insertion de données et de tatouage sont expliqués en détail dans les études [2, 3]. La synchronisation peut être une étape dans le traitement des modèles 3D, elle définit un chemin de sommets (ou de facettes) dans le maillage. Contrairement au cas 2D, il n'y a pas de synchronisation triviale en 3D, mais ces dernières années, c'est un sujet sensible dans plusieurs applications. L'étape de synchronisation dans un système de compression ou de tatouage doit être identique et unique entre les phases de codage et de décodage. En outre si ce système peut subir des attaques ou des dégradations la synchronisation doit être robuste, c'est à dire définir le même ordre pour les deux phases. De plus, dans certains domaines spécialisés (médecine, industrie), la position de sommets et leur connectivité (dans le maillage) ne doivent pas être affectés par ce processus.

Dans cet article, nous proposons une nouvelle méthode de synchronisation robuste basée sur le regroupement de sommets et un schéma de construction itératif. Nous présentons les travaux antérieurs dans la section 2. Ensuite, nous présentons notre méthode dans la section 3. La section 4 expose les résultats de l'approche proposée contre certaines attaques. Enfin, la section 5 conclut le papier et donne quelques pistes de recherches futures.

2 État de l'art

Il existe plusieurs méthodes de synchronisation pour les maillages. Ohbuchi *et al.* [4] ont présenté l'une des premières techniques aveugles qui consiste à dupliquer les arêtes le long d'une bande de triangles, ce qui est visible

(a) *(b)*

Figure 1 – *Deux structures de données uniques appliqués sur Armadillo (19259 sommets) a) ACM, b) Chemin Hamiltonien.*

et facilement détectable. D'autres méthodes de synchronisation basées sur le parcours du maillage ont été réalisée par Mao *et al.* [5] ainsi que par Cayre et Macq [1]. Ces approches dépendent fortement de la connectivité des sommets qui n'est notamment pas robuste aux attaques de remaillage. Cho *et al.* [6] ainsi que Wang *et al.* [7] ont proposés des méthodes basées sur la norme des points, ces méthodes introduisent en général des distorsions visibles et sont dépendantes d'une étape de normalisation. Amat *et al.* [8] ainsi que Tournier *et al.* [9] ont présenté des méthodes de synchronisation basées sur une structure de données unique dans un nuage de points, qui ne dépend que de lui et ne l'affecte pas. La Fig. 1 illustre deux types de structure de données uniques appliqués sur le maillage "Armadillo" [1]. La synchronisation est définie comme la connectivité de l'Arbre Couvrant de poids Minimal (ACM) construit sur les sommets d'un nuage de points en utilisant la distance Euclidienne comme fonction de coût. Un ACM possède les propriétés essentielles que nous recherchons :

– Il est unique.
– Il ne dépend pas de la connectivité du maillage.
– Il ne détériore pas le modèle.
– Il donne un ordre implicite.

Cependant, ces synchronisations ne sont pas assez robustes. Le principal problème de stabilité des ACM est la similitude des distances entre deux sommets dans l'arbre. Une illustration d'un ACM est présenté Fig. 1 (a). L'algorithme de Prim [10] est utilisé pour calculer l'arbre et trouver le voisin le plus proche dans le sous-arbre. Le changement de position des sommets causé par un bruit, peut modifier les distances entre des voisins proches et donc l'ACM. La situation décrite où deux voisins sont environ à la même distance d'un sommet donné du sous-arbre est appelé un "conflit".

Afin de réduire les cas de conflit, nous souhaitons améliorer cette méthode avec un algorithme de partitionnement de donnée. Des travaux ont été effectué sur la segmentation de maillages 3D, comme ceux de Tierny *et al.* [11], mais c'est une approche guidée par la forme. Des méthodes de décimation sont aussi souvent utilisées pour compresser les modèles 3D maillés ou pour les simplifier, comme par exemple les travaux de Lee *et al.* [12]. En règle générale, la compression et la décompression des maillages 3D effectuent le même chemin dans le maillage comme dans la technique *Edgebreaker* de Rossignac [13], ou la méthode guidée par la valence d'Alliez et Desbrun [14] mais de façon non robuste. Nous ne nous sommes pas intéressés non plus à utiliser une décimation basée sur le maillage comme l'a fait Schroeder *et al.* [15], afin de ne pas dépendre de la connectivité.

1. http ://graphics.stanford.edu/

3 La méthode proposée

Comme indiqué dans la section 2, la synchronisation basée sur un ACM [9] n'est pas stable. Néanmoins, ce type de synchronisation est intéressant grâce à ses propriétés. Ces caractéristiques sont la clé pour réaliser une synchronisation robuste et efficace.

La synchronisation basée sur l'ACM n'est pas robuste en raison de l'énorme nombre de sommets qui compose un maillage 3D (plusieurs millions) et de la distance entre eux. La Fig. 2 présente les sommets potentiellement choisis ainsi que le nombre de distances à comparer.

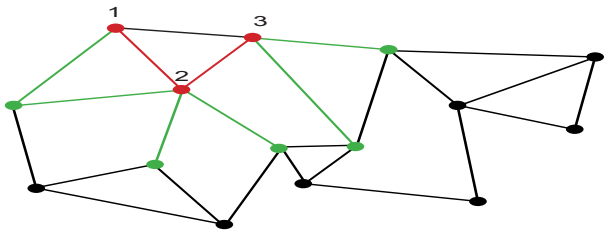


Figure 2 – *Construction d'un ACM, en rouge les sommets et arrêtes ajoutés, en vert les candidats à l'étape suivante.*

Nous proposons d'utiliser la méthode sur moins de sommets. Pour choisir les sommets candidats, nous nous intéressons à un algorithme de décimation qui s'appuie sur le partitionnement dans l'espace du nuage de points. Nous proposons également d'utiliser un chemin Hamiltonien au lieu des ACM pour éviter le plus possible les conflits et réduire la complexité de l'algorithme.

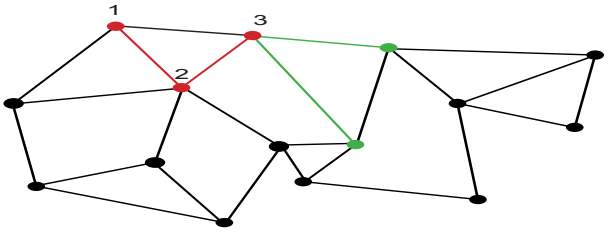


Figure 3 – *Construction d'un chemin Hamiltonien, en rouge les sommets et arrêtes ajoutés, en vert les candidats à l'étape suivante.*

Il apparaît clairement dans la Fig. 3 que la construction d'un chemin Hamiltonien réduit la complexité tout en évitant un grand nombre de comparaisons. Nous remarquons également Fig. 1 (b), que le chemin Hamiltonien a une

structure moins uniforme qu'un ACM. Pour améliorer les performances en termes de robustesse, nous avons réalisé un nouveau type de synchronisation basée sur le regroupement des sommets, une construction itérative et la recherche des plus proches voisins. Notre méthode comporte deux étapes principales. La première, appelée "partitionnement", consiste à diviser le nuage de points 3D en sous-ensemble de points et à calculer pour chacun leurs centroïdes. La deuxième étape consiste à construire le chemin Hamiltonien avec ces centres de façon itérative en utilisant une clé comme point de départ. C'est l'élément secret qui permet d'assurer la sécurité de la synchronisation.

3.1 Partitionnement du nuage de points

Le partitionnement dans l'espace du nuage de points est utilisé comme une étape de pré-traitement dans le but de calculer les sommets intéressants afin de construire le chemin Hamiltonien. Nous commençons par calculer la boîte englobante du nuage de points, puis nous le divisons en un nombre voulus de sous-cubes. A partir de là, nous définissons des groupes qui contiennent tous les sommets inclus dans les sous-cube dont il sont issus. La division est une étape importante pour obtenir des groupes identiques et uniformément répartis, dans le but de récupérer ces mêmes groupes sur un modèle attaqué. La difficulté de cette approche est de définir la taille du sous-cube et le nombre de groupes. Ce nombre de groupe doit être assez grand pour avoir un structure intéressante (ex. insertion de données cachées), mais pas trop grand car il pourrait nuire à la robustesse.

Le volume du sous-cube est calculé comme le volume de la boîte englobante divisé par le nombre de sous-cubes. Puisque nous voulons que notre méthode soit robuste contre un bruit, les paramètres dépendent du niveau de bruit auquel nous voulons être robuste et du nombre de groupes désirés. Comme nous n'avons pas *a priori* d'informations sur la distribution dans l'espace de tous les nuages de points, nous supposons que les sommets sont répartis uniformément dans leur boîte englobante, nous appelons le volume cette boîte vol_B, nb_V le nombre de points et $d(C_i, C_j)$ est la distance entre deux centroïdes C_i et C_j des groupes K_i et K_j. Le nombre de groupes nb_K est donné par :

$$nb_K = \frac{vol_B}{d(C_i, C_j)^3}. \quad (1)$$

A cause de l'hypothèse de distribution uniforme, le nombre de sommets nb_{VK} dans chaque groupe K_i, $i \leq nb_K$ est constant dans le volume d'un sous-cube vol_K, et trivialement nous avons :

$$nb_{VK} = \frac{nb_V}{nb_K}. \quad (2)$$

Dans le cas d'une distribution uniforme, la distance $d(C_i, C_j)$ devrait être supérieure à la variation des positions des sommets produites par un bruit auquel nous voulons que le système soit robuste.

En général, (c.a.d. des distributions non uniformes), l'algorithme produit des sous-cubes vides, chaque sous-cube non vide définit un groupe K_i. Le maillage est subdivisé en groupes et il peut être simplifié en utilisant les centroïdes calculés pour chaque groupe. Certains groupes qui ont un trop petit nombre de sommets ne sont pas conservés parce qu'ils sont trop instables. Lorsque l'étape de décimation est finie, nous obtenons un ensemble de sommets S tel que $S \in C$ et nous pouvons limiter la construction du chemin Hamiltonien au graphe complet $G = (S, A)$. Toutefois, nous utilisons le lien entre chaque sommet de S et son groupe associé pour permettre la transformation du graphe G par l'adaptabilité de notre méthode de construction présentée dans la section 3.2.

Afin de lier les groupes les uns aux autres, on considère qu'un groupe est voisin à un autre si ils sont voisins sur le maillage. Dans la section 3.2, nous expliquons qu'il est important pour un groupe de connaître ses voisins pour caractériser les conflits qui pourraient survenir. L'algorithme de mise à jour du voisinage est exécuté à chaque modification du graphe G.

3.2 Construction itérative du chemin

Le chemin Hamiltonien est construit sur G de façon itérative en ajoutant le plus proche voisin du dernier sommet sélectionné. La Fig. 4 illustre la manière dont la construction du chemin s'effectue. Pour préserver la sécurité, la clé secrète donne le premier groupe K_0 et son centroïde $C_0 \in S$ est la graine. A chaque itération, pour éviter les conflits, l'algorithme vérifie si il existe un autre candidat à environ la même distance. Nous utilisons un seuil γ pour déterminer si il y a conflit entre deux distances. Il dépend du déplacement moyen possible des sommets, dû au bruit ou à une attaque, contre lequel le système doit être robuste. A l'itération i, C_j est le plus proche voisin de C_i et C_k est un autre sommet potentiellement choisi. Alors, pour $k \neq i, j$, et C_k n'appartenant pas au chemin déjà construit, il faut que :

$$\begin{aligned} &d(C_i, C_j) < d(C_i, C_k), \\ &|d(C_i, C_j) - min(d(C_i, C_k))| > \gamma. \end{aligned} \quad (3)$$

Dans ce cas, il n'y a pas de conflit entre deux sommets, le sommet choisi devient la graine, et ainsi de suite. Sinon, nous considérons deux cas :

1. Les deux sommets appartiennent à deux groupes voisins.
2. Les deux sommets n'appartiennent pas à deux groupes voisins.

Ces propriétés nous ont conduit à introduire une application multi-résolution. Dans cette situation, un changement d'échelle génère un nouveau comportement qui permet d'éviter les conflits. Pour changer d'échelle dans le premier cas, nous fusionnons les deux groupes qui contiennent les candidats. Dans le second cas, nous divisons ces groupes.

Fonction de fusion. La fusion de deux groupes K_i, K_j implique l'union de ces groupes dans un nouveau $K_k = K_i \cup K_j$, et le calcul du nouveau centroïde. Puis,

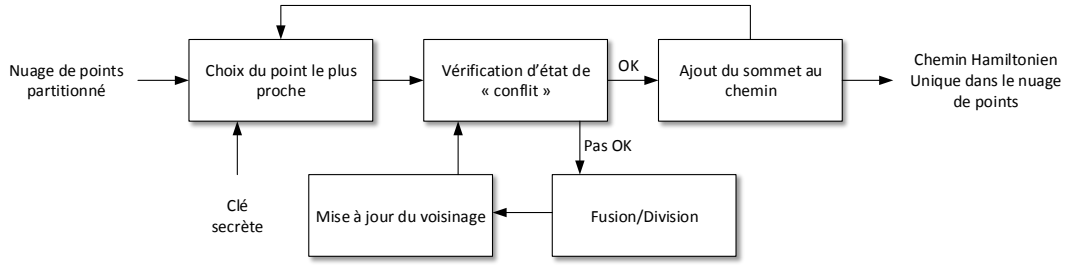


Figure 4 – *Schéma de la construction itérative du chemin Hamiltonien.*

afin de maintenir le voisinage, on note $\mathcal{N}(K_i)$ l'ensemble des voisins de K_i, alors :

$$\mathcal{N}(K_k) = (\mathcal{N}(K_i) - \{K_j\}) \cup (\mathcal{N}(K_j) - \{K_i\}).$$

La fonction de fusion évite les conflits en supprimant les candidats et en en créant un nouveau qui est la moyenne des autres. Dans ce cas, si le seuil γ est beaucoup plus grand que la distance moyenne entre deux centroïdes, il y a un risque de sur-fusionner. Inversement, si le seuil γ est trop petit, il n'y aura pas de fusions.

Fonction de division. Dans cette étape la fonction divise les deux groupes qui sont en conflit. Cela évite d'avoir à faire un choix, évitant ainsi les erreurs lors de la reconstruction. La division d'un groupe consiste à diviser l'original en deux groupes plus petits. Contrairement à la fusion qui est assez simple, nous devons définir la procédure de division. Afin d'éviter les conflits après l'étape de division, les distances entre la graine C_i et tous les groupes créés ne devraient pas être trop similaires. Nous appelons $K_{k,1}$, $K_{k,2}$ respectivement $K_{j,1}$, $K_{j,2}$ les nouveaux groupes créés par la division K_k et K_j. Si le centroïde $C_{j,1}$ de $K_{j,1}$ est le prochain sommet choisi, et si :

$$\alpha = min(d(C_i, C_{j,2}), d(C_i, C_{k,1}), d(C_i, C_{k,2}))$$

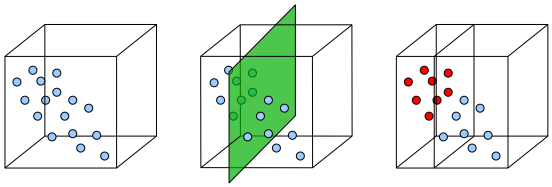
les centroïdes des groupes doivent vérifier la condition :

$$d(C_i, C_{j,1}) + \gamma < \alpha. \quad (4)$$

Si la condition est vérifiée, alors il n'y a plus de conflit et l'algorithme passe à l'étape suivante de mise à jour du voisinage, comme illustré Fig. 4. Néanmoins, si l'équation (4) est n'est pas respectée à cause d'un seuil γ trop grand, après avoir divisé les groupes, notre système risque de boucler infiniment. Quand un groupe est le résultat d'une division, s'il est de nouveau en conflit, le seuil appliqué est diminué de moitié (c.à.d. $\gamma/2$). Afin de satisfaire l'équation (4), nous devons trouver le plan qui maximise la différence :

$$diff = |d(C_i, C_{j,1}) - d(C_i, C_{j,2})|. \quad (5)$$

Il est défini dans l'ordre suivant : on tente d'abord de diviser le long de l'axe X, puis l'axe Y et enfin l'axe Z. L'algorithme s'arrête lorsque la distribution des sommets entre les groupes se situe entre 40% et 60% et la division se fait le long de l'axe approprié. Nous devons définir un ordre fixe, au lieu de chercher le meilleur plan, pour avoir



(a) *(b)* *(c)*

Figure 5 – *a) Groupe original, b) Division sur le plan d'un axe, c) Les deux groupes créés.*

un algorithme déterministe, et obtenir de meilleurs résultats.

Nous pouvons maintenant introduire un mécanisme illustré dans la Fig. 5, qui nous permet de construire le plan de séparation d'un groupe K_i. Le vecteur normal $\vec{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$ est calculé en fonction du choix de l'axe. Le centroïde de K_i donne le vecteur $\vec{V} = \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix}$. Il permet alors de définir le vecteur normal comme : $\vec{n} = \begin{bmatrix} x_v - x_a \\ y_v - y_a \\ z_v - y_a \end{bmatrix}$, où $\vec{A} = \begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix}$ est un vecteur dirigé par l'axe choisi. Par exemple, si X est l'axe choisi et $maxD$ est la plus longue distance dans le modèle 3D, alors $\vec{A} = \begin{bmatrix} x_a \\ 0 \\ 0 \end{bmatrix}$, où $x_a \gg maxD$.

Alors, pour tous les sommets $w_k \in K_i$ représentés par $W_k = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}$, l'algorithme contrôle de quel côté du plan le sommet est situé, avec :

$$\begin{aligned} pos \quad &= n_1(x_k - x_v) + n_2(y_k - y_v) + n_3(z_k - z_v), \\ &\begin{cases} si\ pos\ \leq 0\ alors\ w_k \in K_{i,1} \\ sinon\ w_k \in K_{i,2}. \end{cases} \end{aligned} \quad (6)$$

L'étape suivante consiste à calculer le centroïde des nouveaux groupes. Les nouveaux groupes sont alors connectés aux autres en recalculant le voisinage et en mettant à jour le graphe G.

Après une fonction de division ou de fusion, il faut vérifier les conditions données par l'équation (4). Comme indiqué Fig. 4, l'algorithme boucle sur la même graine. Grâce à la

capacité d'adaptation du seuil, il ne peut pas boucler sans fin. Le résultat est un chemin Hamiltonien construit de façon déterministe.

4 Résultats expérimentaux

La méthode proposée a été testée sur plusieurs modèles 3D maillés. Nous avons analysé notre méthode de synchronisation sur quatre modèles à mailles triangulaires normalisées (en fonction de la taille du volume englobant). Le tableau 1 illustre le comportement de notre application. La deuxième colonne indique la taille en nombre de sommets du maillage. La troisième colonne indique le nombre de groupes initial non vide (GI). La quatrième colonne indique le nombre de groupes dans le chemin Hamiltonien unique (CHU). Les colonnes suivantes montrent le nombre de fusions et de divisions. Par l'équation (1), nous obtenons un nombre de groupe d'environ 640. Ce nombre étant calculé sur la base d'hypothèses fortes (distribution uniforme par exemple) il n'est pas retrouvé dans les expérimentations. La forme et la densité du modèle implique la formation de groupes vides.

Tableau 1 – *Résultats pour différents modèles.*

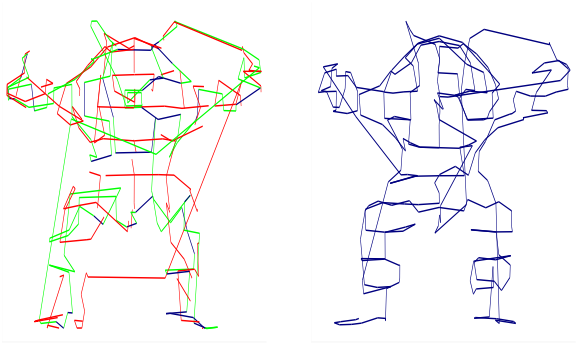
Modèles	Taille	#GI	#G du CHU	#F	#D
Bunny	34.834	88	98	0	5
Horse	48.485	94	104	0	5
Armadillo	172.974	229	164	3	19
Blade	220.559	291	673	2	192

Pour l'expérimentation nous voulons résister à un bruit Gaussien avec $\sigma = 10^{-2}$. Nous savons que presque toutes les valeurs sont comprises entre trois fois l'écart type $3.\sigma$, excepté pour certaines valeurs aberrantes. Nous choisissons un $\sigma = 10^{-1}$ beaucoup plus élevé que $3.\sigma$, car dans un cas réel nous n'avons pas une distribution uniforme des points. Ensuite, nous calculons le nombre de sommets de chaque groupe, soit 270 pour le modèle "Armadillo". En pratique, les groupes ont en moyenne 800-1000 sommets. Pour comparer les résultats et montrer l'influence du nombre de sommets et de la forme des modèles, nous les comparons avec ce paramètre fixé à 200. Logiquement, les plus grands modèles produisent plus de groupes. Le nombre de fusions et de divisions augmentent avec la taille du modèle en raison du plus grand nombre de groupes créés. Nous remarquons qu'il y a moins de fusions que de divisions.

Un des principaux défis est de trouver la taille qui maximise le nombre de sommets dans le chemin Hamiltonien tout en maintenant sa robustesse.

Tableau 2 – *Pourcentage des arcs communs entre les chemins construits sur le modèle 3D d'origine et sur le bruité pour un bruit Gaussien avec* $\sigma = 10^{-6}$.

Modèles	Bunny	Horse	Armadillo	Blade
MN	77%	80%	20%	27%
MP	100%	100%	100%	100%



(a) *(b)*

Figure 6 – *a) Construction d'un chemin Hamiltonien à partir d'une décimation naïve, b) Construction d'un chemin Hamiltonien avec notre approche.*

Le nombre de groupes obtenus n'est pas prévisible. Nous pouvons expliquer cela par la forme et la densité du modèle. Plus le modèle est dense, plus le facteur de seuil va conduire à des conflits.

Le tableau 2 compare les résultats de la méthode proposée "MP" avec une naïve "MN" pour une taille de groupe moyenne de 200 points et un bruit Gaussien avec $\sigma = 10^{-6}$. La méthode naïve consiste à construire chemin Hamiltonien sans notre étape de vérification d'état de conflit, sur un ensemble de sommets obtenus par une décimation classique définie par Garland [16]. Ces résultats montrent que notre méthode est sûre pour ce genre de bruit, alors que la méthode naïve n'est jamais juste. D'autre part, comme les modèles "Bunny" et "Horse" sont plus petits en nombre de points, le chemin Hamiltonien construit par la méthode naïve est moins détérioré que ceux construits sur les modèles plus grands.

Afin d'illustrer le contenu du tableau 2, nous avons appliqué un bruit Gaussien avec $\sigma = 10^{-6}$ sur l'objet "Armadillo", puis nous avons calculé le chemin Hamiltonien en utilisant dans un premier temps une décimation naïve sur l'original et le bruité Fig. 6 (a), et ensuite en utilisant notre algorithme Fig. 6 (b). Les arcs en bleu sont les arcs communs, ceux en rouge sont ceux du modèle 3D d'origine et en vert ceux du bruité. Le pourcentage des arcs communs n'est que de 20% pour la méthode naïve alors qu'il est de 100% avec notre méthode.

Plus généralement nous avons analysé les quatre modèles, les résultats apparaissent sur le graphe de la Fig. 7 qui montre l'influence d'un bruit Gaussien sur ces modèles. Nous comparons notre méthode avec la méthode naïve "DN". La robustesse est définie comme le pourcentage des arcs communs entre le chemin Hamiltonien calculé sur un modèle et le chemin Hamiltonien calculé sur le même modèle bruité. Les courbes montrent que pour un bruit Gaussien avec σ inférieur à 10^{-4}, le chemin Hamiltonien est robuste contre les attaques qui déplacent les sommets. Nous voulions assurer la robustesse contre un bruit de l'ordre de $\sigma = 10^{-2}$, mais nos résultats sont théoriques. Pourtant, pour un bruit plus important, on voit qu'il y a encore plus de 50% d'arcs en communs. Comme mentionné précédem-

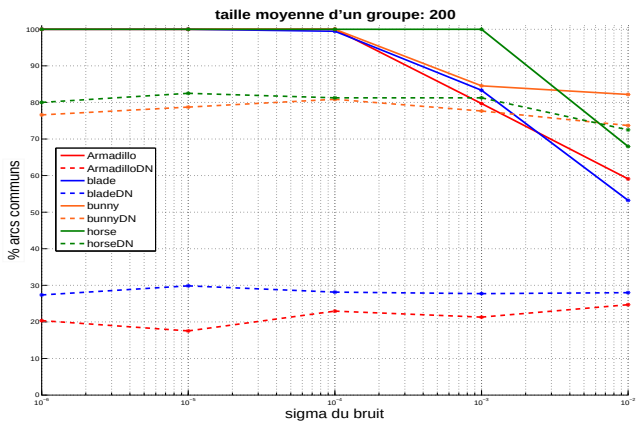


Figure 7 – *Stabilité des chemins contre un bruit Gaussien.*

ment, les paramètres doivent être définis : la taille moyenne d'un groupe est fixé à 200 sommets, ce qui est assez petit pour avoir un bon nombre de groupes pour les petits modèles comme les modèles "Bunny" et "Horse". Mais il est trop petit pour assurer la robustesse des modèles plus grands. Ces courbes montrent que nous devons faire des compromis. Cependant, notre méthode est toujours mieux que la méthode naïve. Un autre résultat intéressant est que notre méthode est clairement robuste contre les attaques de remaillage, car elle n'est pas basé sur la connectivité du maillage.

5 Conclusion

Dans cet article, nous avons proposé une nouvelle méthode robuste et évolutive pour décrire un modèle 3D par un chemin Hamiltonien unique. En effet, notre système itératif permet d'éviter certains conflits. Cette méthode est efficace contre une attaque de type bruit Gaussien et nos expériences le prouvent. Elle peut être utilisée comme méthode de synchronisation pour l'insertion de données cachées, car elle possède de bonnes propriétés : elle est aveugle et elle ne modifie pas le maillage.
Pour l'améliorer, nous voulons mettre en place un ordre dans le chemin, pour éviter certaines attaques comme le rognage. Nous envisageons aussi d'autres attaques et d'autres méthodes de partitionnement de l'espace. Enfin, nous voulons améliorer les règles de choix des sommets au cours de la construction du chemin Hamiltonien. L'objectif de nos futurs travaux, est d'améliorer notre méthode pour être robuste à d'autres types d'attaques et d'utiliser ces résultats pour obtenir une bonne synchronisation et proposer une méthode de tatouage.

Références

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

[2] P. Rondao Alface et B. Macq. From 3d mesh data hiding to 3d shape blind and robust watermarking : A survey. Dans YunQ. Shi, éditeur, *Transactions on Data Hiding and Multimedia Security II*, volume 4499 de *Lecture Notes in Computer Science*, pages 91–115. Springer Berlin Heidelberg, 2007.

[3] K. Wang, G. Lavoue, F. Denis, et A. Baskurt. A comprehensive survey on three-dimensional mesh watermarking. *Multimedia, IEEE Transactions on*, 10(8) :1513 –1527, dec. 2008.

[4] R. Ohbuchi, H. Masuda, et M. Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. *Selected Areas in Communications, IEEE Journal on*, 16(4) :551–560, may 1997.

[5] X. Mao, M. Shiba, et A. Imamiya. Watermarking 3d geometric models through triangle subdivision. pages 253–260, 2001.

[6] J.-W. Cho, R. Prost, et H.-Y. Jung. An oblivious watermarking for 3-d polygonal meshes using distribution of vertex norms. *Signal Processing, IEEE Transactions on*, 55(1) :142 –155, jan. 2007.

[7] K. Wang, G. Lavoué, F. Denis, et A. Baskurt. Robust and blind mesh watermarking based on volume moments. *Computers & Graphics*, 35(1) :1–19, 2011.

[8] 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.

[9] N. Tournier, W. Puech, G. Subsol, et J.-P. Pedeboy. Finding robust vertices for 3d synchronization based on euclidean minimum spanning tree. *Proc. SPIE 7864 Three-Dimensional Imaging, Interaction, and Measurement*, pages 78640W–78640W–7, 2011.

[10] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36 :1389–1401, 1957.

[11] J. Tierny, J.-P. Vandeborre, et M. Daoudi. Topology driven 3D mesh hierarchical segmentation. Dans *IEEE International Conference on Shape Modeling and Applications (Shape Modeling International 2007)*, Lyon, France, June 13-15 2007.

[12] C. H. Lee, A. Varshney, et D. W. Jacobs. Mesh saliency. *ACM Trans. Graph.*, 24(3) :659–666, 2005.

[13] J. Rossignac. Edgebreaker : Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1) :47–61, Janvier 1999.

[14] P. Alliez et M. Desbrun. Valence-driven connectivity encoding for 3d meshes. *Computer Graphics Forum*, 20(3) :480–489, 2001.

[15] W. J. Schroeder, J. A. Zarge, et W. E. Lorensen. Decimation of triangle meshes. *SIGGRAPH Comput. Graph.*, 26(2) :65–70, Juillet 1992.

[16] M. Garland. *Quadric-based polygonal surface simplication.* Thèse de doctorat, School of Computer Science, Carnegie Mellon University, 1999.