Journées du Groupe de Travail en Modélisation Géométrique 2013, Marseille

Transport optimal pour la reconstruction robuste de formes à partir de nuages de points

Julie Digne [1], Pierre Alliez[2] et David Cohen-Steiner[2]

[1]LIRIS - Université Lyon 1
[2]Inria Sophia Antipolis - Mediterranée

Résumé

Résumé: Notre approche consiste à considérer le nuage de points en entrée comme une mesure discrète (une distribution de masses), et à construire une approximation par une mesure continue (et constante par morceaux) sur les faces d'un complexe simplicial. La distance entre les deux mesures est calculée par une approximation du transport optimal obtenue par programmation linéaire, et le complexe simplicial est obtenu par décimation et optimisation d'une triangulation de Delaunay initialisée avec un sous-ensemble des points en entrée. La distance utilisée est robuste à la fois au bruit et aux données aberrantes, et préserve les arêtes vives et les bords des formes à reconstruire. Cette distance peut également servir comme outil de post-traitement sur des surfaces lisses reconstruites avec des méthodes par fonction implicite.
Abstract: We present a robust and feature-capturing surface reconstruction method based on optimal transport that turns an input point set into a low triangle-count simplicial complex. Our approach starts with a simplicial complex filtered from a 3D Delaunay triangulation of the input points. This initial approximation is iteratively simplified based on an error metric that measures, through optimal transport, the distance between the input points and the current simplicial complex—both seen as mass distributions. Our approach is shown to exhibit both robustness to noise and outliers, as well as preservation of sharp features and boundaries. Our new feature-sensitive metric between point sets and triangle meshes can also be used as a post-processing tool that, from the smooth output of a reconstruction method, recovers sharp features and boundaries present in the initial point set.

1. Introduction

La reconstruction de surfaces est un problème complexe pour lequel il n'existe pas une solution unique, du fait de la variété des données d'entrée. De nombreuses méthodes ont été proposées [Ame99, KBH06], mais la reconstruction de surfaces préservant les arêtes vives et robustes au bruit de mesure reste encore un problème difficile. Dans cet article nous présentons une méthode de reconstruction qui simplifie un complexe simplicial (éventuellement non-variété) construit en triangulant le nuage de point initial. La simplification est guidée par une métrique de transport optimal entre le nuage de points et le complexe simplicial courant. La reconstruction hérite des qualités de la métrique: robustesse au bruit et aux points aberrants. Appliquée à la sortie d'une méthode de reconstruction lisse ou semi-lisse, notre métrique peut être utilisée pour retrouver les arêtes vives et bords à travers un processus de déplacement des sommets.

2. état de l'art

2.1. Reconstruction de surface

Une approche habituelle pour la reconstruction de surface à partir de nuages de points imparfaits consiste à débruiter et filtrer les point aberrants. Cela demande souvent un ajustement des paramètres par l'utilisateur. Des méthodes automatiques comme les méthodes spectrales [KSO04, WCS05, ACSTD07] et graph-cut [HK06, LPK09] sont robustes mais adaptées à la reconstruction de surfaces lisses et sans bord. Une série de méthodes plus récentes sont basées sur des normes robustes et la parcimonie [LCOLTE07, HLZ*09, ASGCO10]. Une méthode interpolante mais néanmoins robuste au bruit a été proposée par Digne et al. [DMSL11] à travers l'introduction d'un espace-échelle.

Les méthodes préservant les arêtes vives sont usuellement basées sur des représentation implicites qui approchent ou interpolent les points initiaux. Les arêtes vives sont par ex-

emple capturées par des fonctions de base adaptées localement dans [DTS01]. Des méthodes Moving Least Squares ([AA06], [OGG09]) ont également été proposées pour recouvrer les arêtes vives. Cependant aucune de ces formulations ne permet de retrouver de vraies arêtes vives, les reconstructions sont toujours semi-lisses, c'est à dire arrondies à divers degrés. De plus, les détections d'arêtes sont uniquement locales ce qui donne des arêtes fragmentées dans la reconstruction.

Une autre façon de détecter les arêtes locales consiste à segmenter localement les normales estimées, et à faire autant de régressions de quadriques que de segments trouvés [OBA*03] ou à segmenter les voisinages par croissance de région [FCOS05]. D'autres approches qui donnent aussi des détections locales et donc des arêtes fragmentées ont été proposées [LCOL07, GWM01, PP09]. Pour réduire ces défauts de fragmentation, une approche consiste à extraire de longues lignes de crêtes. [PKG03] utilise une approche multi-échelle pour détecter les points d'intérêt et construire un graphe d'arêtes vives. La simplification de formes a aussi été abordée dans [BC01] qui mêle également reconstruction et simplification.

Notre méthode adopte une autre approche: la forme est reconstruite à travers une simplification itérative d'un complexe simplicial construit à partir du nuage de points. Pour obtenir la robustesse au bruit et aux points aberrants, la métrique guidant la reconstruction est formulée comme une distance de transport optimal entre le nuage de points et le maillage reconstruit. Cet article est une version courte de [DCSA*13].

2.2. Transport Optimal

Le problème du transport d'une mesure sur une autre comme façon de quantifier leur similarité est un problème mathématique bien identifié. Pour deux mesures μ et ν définies sur $\mathbb{R}^3$ et de masse totale égale, le transport optimal L_2 de μ à ν consiste à trouver un plan de transport π réalisant l'infimum:

$$\inf\left\{ \int_{\mathbb{R}^3} \|x-y\|^2 \mathrm{d}\pi(x,y) \,\middle|\, \pi \in \Pi(\mu,\nu) \right\},$$

où $\Pi(\mu,\nu)$ est l'ensemble de tous les plans de transport possibles entre μ et ν [Vil10].

Pour résumer, un transport optimal π est un déplacement de chaque masse infinitésimale de la mesure d'entrée μ (ici le nuage de points) sur la mesure cible ν (ici le complexe simplicial). Cette formulation est particulièrement adaptée pour comparer des mesures 1D, comme des histogrammes, pour transporter des couleurs entre des images [RAGS01, RDG10].

Pour des applications de plus grandes dimensions [RTG00, RPC10, PFR11], le transport optimal est beaucoup plus compliqué: la résolution de problèmes de transport optimal requiert souvent un programme linéaire (LP) par exemple pour la comparaison de surface [LD10] et l'interpolation de déplacement [BvdPPH11].

Contributions. Notre formulation est également basée sur une formulation LP. Cependant, une grande différence est que la mesure cible ν n'est pas donnée mais est une partie du résultat. Plus précisément, nous cherchons le complexe simplicial de taille donnée par l'utilisateur qui minimise le coût de transport de la mesure de départ (le nuage de points) sur les simplexes du complexe.

De Goes et al. [dGCSAD11] ont proposé un algorithme pour reconstruire et simplifier des formes 2D également basé sur du transport optimal. Cependant notre approche diffère sensiblement de cette première proposition en plusieurs aspects: leur solution de transport optimal pour des points et des arêtes peut être calculée en forme close, mais une telle forme close n'existe pas pour les surfaces. Nous utiliserons donc une version discretisée du transport. Chaque point est également assigné au sommet ou arête le plus proche, or ce schéma simpliste donne des plans de transport largement sous-optimaux. Notre formulation discrète combinée à un solveur de programme linéaire donne une meilleure approximation du plan et coût de transport. Leur méthode nécessite de plus un plongement valide des triangulations 2D qui est maintenu par une séquence de bascules d'arêtes. Malheureusement une telle procédure n'est pas généralisable en 3D. Notre méthode abandonne donc la contrainte du plongement et ne reconstruit qu'un complexe simplicial, choisi initialement comme un sous-ensemble d'une triangulation de Delaunay 3D.

2.3. Algorithme

L'algorithme reconstruit une surface à partir d'un nuage de points par une simplification gloutonne d'un complexe simplicial 3D. Ce complexe est initialisé comme un sous-ensemble d'une triangulation de Delaunay des points initiaux. La décimation est ensuite réalisée par des fusions de demi-arêtes. L'arête choisie pour fusion est celle qui minimise l'augmentation de l'erreur d'approximation, erreur calculée comme une distance de transport optimal entre le nuage de points (vue comme des mesures de Dirac) et une mesure constante par facette définie sur tout le complexe simplicial. Cet algorithme est résumé dans la figure 1, et les étapes principales sont montrées sur la figure 1.

3. Formulation de la reconstruction comme un problème de transport

Dans notre formulation le problème de la reconstruction est de passer d'un nuage de points S à un complexe simplicial $\mathcal{C}$. Le nuage de points contient N points $(p_i)_{i=1\cdots N}$ et chaque point a une masse m_i qui peut par exemple refléter la confiance en la mesure (par défaut, tous les points ont la même masse). Le nuage de points ainsi que le complexe simplicial sont vus comme des distributions de masse, où la mesure sur $\mathcal{C}$ est constante par simplexe et potentiellement nulle.

Algorithm 1: Algorithme général.

Entrée: Nuage de points $\mathcal{S}$, nombre de sommets souhaité V.
Sortie : Complexe simplicial $\mathcal{C}$ avec V sommets.
Construire la triangulation de Delaunay 3D $\mathcal{T}$ à partir de $\mathcal{S}$;
Calculer le coût de transport de $\mathcal{S}$ aux facettes de $\mathcal{T}$;
Construire le complexe simplicial $\mathcal{C}$ à partir des facettes de $\mathcal{T}$ de mesure non nulle;
Décimer $\mathcal{C}$ jusqu'à obtenir le nombre de sommets souhaité V;
Filtrer les facettes de $\mathcal{C}$ en seuillant sur la densité de masse.

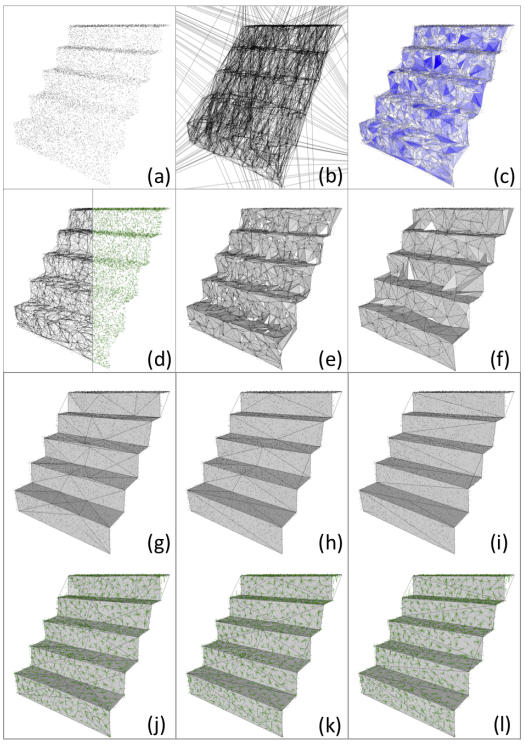

Figure 1: *Etapes de l'algorithme: (a) Nuage de points initial; (b) Triangulation de Delaunay 3D d'un sous-ensemble aléatoire de* 10% *du nuage initial; (c) Complexe simplicial initial construit à partir des facettes de la triangulation de Delaunay de mesure non nulle; (d) Plan de transport initial assignant des échantillons à des centres de cellules (flêches vertes); (e-f) étapes de décimation intermédiaires; (g-i) Reconstruction à* $100, 50, 20$ *sommets respectivement; (j-l) reconstruction à* $100, 50, 20$ *sommets respectivement.*

3.1. Discrétisation

Le coût de transport entre le nuage de points initial $\mathcal{S}$ et $\mathcal{C}$ le complexe simplicial est approché en réalisant une quadrature. Nous commençons par définir un ensemble $\mathcal{B}$ de cellules (des petite régions de $\mathcal{C}$) sur les facettes de $\mathcal{C}$. Nous ajoutons à cet ensemble une cellule par sommet de $\mathcal{C}$ qui

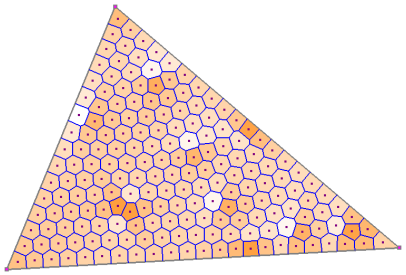

Figure 2: *Cellule d'une facette. Les cellules d'une facette sont définies comme les cellules d'un diagramme de Voronoi centré. Leurs centres de masse sont représentés par des points rouges. Leurs capacités (proportionnelles à leurs aires) sont représentées par une rampe de couleur.*

seront utiles dans les cas de points aberrants. Ces cellules sont utilisées pour évaluer le coût de transport comme une somme de distances carrées entre les points de $\mathcal{S}$ et les centres de masse des cellules de $\mathcal{C}$.

Chaque sommet de $\mathcal{C}$ est considéré comme le centre de sa propre cellule. Chaque facette est échantillonnée par des cellules grâce à un diagramme de Voronoi centré (CVT) (Figure 2). Le nombre de cellules par facette est calculé selon une quadrature donnée par l'utilisateur. Dans toutes les expériences nous utilisons une quadrature de 200 cellules par unité d'aire. Pour compenser la légère non-uniformité de la répartition des cellules, on donne à chaque cellule b_j une capacité c_j égale au ratio de son aire sur l'aire du triangle. La masse reçue ne pourra donc excéder $c_j \times$ masse reçue par le triangle complet. Dans le cas des sommets, la capacité est mise à 1, et la masse reçue n'est pas contrainte (puisqu'il n'y a qu'une seule cellule par sommet).

3.2. Formulation comme problème linéaire

La précédente discrétisation est utilisée pour calculer le coût de transport optimal à travers un problème linéaire. Par la suite nous notons $(\sigma_j)_{j=1\cdots L}$ les simplexes de $\mathcal{C}$ et les centres de masse des cellules comme $(b_k)_{k=1\cdots M}$ où L et M sont le nombre de simplexes et de cellules respectivement. La capacité d'une cellule b_j est notée c_j et $s(j)$ est l'index du simplexe contenant la cellule b_j. m_{ij} est la quantité de masse transportée du point initial $p_i \in \mathcal{S}$ à la cellule b_j (figure 3). Le plan de transport entre $\mathcal{S}$ et $\mathcal{C}$ est donc un ensemble $N \times M$ de variables m_{ij} telles que:

$$\forall ij : \; m_{ij} \geq 0, \tag{1}$$

$$\forall i : \; \sum_j m_{ij} = m_i, \tag{2}$$

$$\forall j_1, j_2 \text{ s.t. } s(j_1) = s(j_2) : \; \frac{\sum_i m_{ij_1}}{c_{j_1}} = \frac{\sum_i m_{ij_2}}{c_{j_2}}, \tag{3}$$

où l'équation 2 garantit que la mesure entière du nuage de points est transportée sur $\mathcal{C}$, l'équation 3 garantit que la

mesure est uniforme sur chaque facette de $\mathcal{C}$. Un *plan de transport optimal* est alors défini comme un plan de transport π qui minimise son coût de transport associé:

$$cost(\pi) = \sum_{ij} m_{ij} \|p_i - b_j\|^2.$$

Ainsi, trouver un plan de transport minimisant le coût de transport est un problème linéaire par rapport aux variables m_{ij}. Il faut remarquer que le nombre de cellules, leurs positions et les carrés des distances entre les centres de masse et les points sont précalculés. Pour garantir la contrainte d'uniformité (Eq. 3) simplement, L variables l_i additionnelles sont introduites (une par simplexe σ_i). Elles correspondent à la densité de la mesure cible du simplexe correspondant. La formulation finale du problème est donc:

Minimiser $\sum_{ij} m_{ij} \|p_i - b_j\|^2$
par rapport aux variables m_{ij} et $l_{s(j)}$ sous les contraintes:

$$\begin{cases} \forall i : \sum_j m_{ij} = m_i \\ \forall j : \sum_i m_{ij} = c_j \cdot l_{s(j)} \\ \forall i,\ j : m_{ij} \geq 0,\ l_{s(j)} \geq 0 \end{cases}$$

3.3. Relaxation locale

Résoudre directement la formulation ci-dessus est très coûteux en calcul car il faut instancier une matrice dense de taille $(M \times N + L) \times (M + N)$. Par exemple, calculer le coût de transport entre le nuage de points initial de $2,100$ échantillons et un complexe simplicial de 782 simplexes et 7300 cellules implique 15 millions de variables et 9000 contraintes. Les solveurs actuels ne permettent pas de traiter des problèmes de cette taille.

Afin d'améliorer le passage à l'échelle on opère une relaxation locale. celle-ci explore seulement un sous-espace de l'espace des solutions du problème global à travers des résolutions de problèmes LP sur des voisinages locaux. Il n'y a alors pas de garantie de convergence au minimum global, mais le minimum atteint est en pratique suffisant pour notre problème. Tout d'abord, un plan de transport trivial est calculé: chaque point est assigné au sommet de $\mathcal{C}$

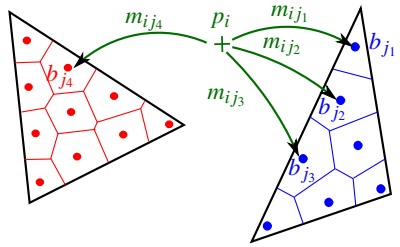

Figure 3: *Plan de transport d'un point p_i du nuage. m_{ij} représente le transport de sa masse m_i à la cellule b_j de $\mathcal{C}$.*

Algorithm 2: Relaxation par résolutions locales.

Entrée: Complexe simplicial $\mathcal{C}$, nuage de points $\mathcal{S}$, seuil ε
Sortie : Plan de transport $\pi = \{m_{ij}\}$
for $p_i \in \mathcal{S}$ **do**
  Transporter p_i au plus proche sommet $v \in \mathcal{C}$;
`new_cost` $\leftarrow 0$;
repeat
  for $\sigma_j \in \mathcal{C}$ **do**
    `old_cost` $\leftarrow$ `new_cost`;
    Collecter le voisinage $\mathcal{N}$ de la facette σ_j;
    Collecter les points et masses partielles $\{p_i, \widetilde{m}_i\}$ qui se transportent sur $\mathcal{N}$;
    Résoudre le programme linéaire pour trouver le plan de transport optimal de $(p_i, \widetilde{m}_i)$ sur les cellules de $\mathcal{N}$;
    Mettre à jour le plan de transport π et le coût `new_cost`;
  $\delta =$ `new_cost` $-$ `old_cost`;
until $\delta \leq \varepsilon$;

le plus proche. Ce plan de transport satisfait les conditions car la contrainte d'uniformité est trivialement satisfaite sur les sommets, mais il est clairement sous-optimal. Des optimisations locales vont ensuite faire uniquement décroitre le coût de transport ou le laisser inchangé. Le coût de transport trouvé est donc une borne supérieure du coût de transport global. En pratique utiliser le $1-$*voisinage* local donne des résultats satisfaisants. Grâce à cette approximation nous décrivons maintenant comment s'en servir pour la reconstruction de surface.

4. Reconstruction par simplification

4.1. Initialisation

Le processus de reconstruction commence par choisir aléatoirement un sous-ensemble du nuage de points d'entrée $\mathcal{S}$ et calcule D, une triangulation de Delaunay 3D. Nous construisons ensuite un complexe simplicial $\mathcal{C}$ à partir d'un sous-ensemble des facettes de D. Pour sélectionner ces facettes, la relaxation locale (Algorithm 2) donne une estimation du coût de transport sur chaque facette. Le voisinage utilisé pour la relaxation locale est dans ce cas centré sur une facette. Il s'agit de l'ensemble des facettes et vertices des deux tétraèdres (au plus) adjacents à la facette centrale.

Une optimisation est ensuite réalisée en résolvant localement sur tous les voisinages de la triangulation, puis en itérant jusqu'à obtenir une variation de coût en dessous d'un seuil (par exemple 10^{-5}). En pratique le coût décroit trés vite, et il suffit de parcourir tous les voisinages une dizaine de fois. Finalement seules les facettes sur lesquelles une masse non-nulle est transportée sont conservées. Si le sous-

ensemble des points utilisé pour construire D est trop grand cette premiére étape échouera à avoir un nombre de facettes de départ suffisamment représentatif de la forme.

4.2. Décimation

A partir du complexe initial $\mathcal{C}$, la simplification est itérée à travers une décimation gloutonnee basée sur des fusions de demi-arêtes. L'arête suivante à fusionner est celle qui induit l'accroissement minimum du coût de transport global. Chaque fusion de demi-arête est donc simulée et la variation de coût de transport induite est évaluée comme différence Δ de coût de transport avant et aprés fusion (en se restreignant à la fermeture des simplexes de 1-ring, puisque la fusion ne change que ces simplexes). Il suffit donc de maintenir à jour une queue de priorité des arêtes à fusionner triées par Δ croissant et de fusionner la première arête. Ceci est également coûteux car après une fusion il faut mettre à jour un grand nombre de simulations d'arêtes. On se contente donc d'une stratégie encore plus simple: une stratégie *choix multiples* qui consiste à tirer aléatoirement un sous-ensemble d'arêtes et de fusionner celle ayant le plus petit Delta, en re-simulant si nécessaire les fusions des arêtes dont le Δ ne serait pas à jour. Ce processus de décimation est résumé par la figure 3.

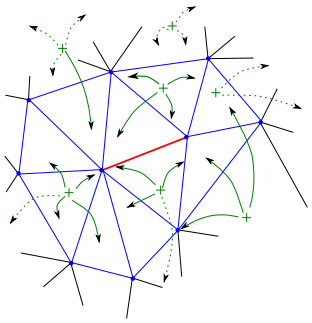

Figure 4: *Voisinage local d'une arête. Par soucis de clarté, un voisinage variété d'une arête est représenté. Les simplexes du voisinage de l'arête (en rouge) sont représentés en bleu. Les points du nuage sont représentés en vert. L'algorithme résout seulement les mesures transportées vers le voisinage (lignes vertes solides) et non celles transportées à l'extérieur (lignes vertes pointillées).*

4.3. Déplacement de sommets

Jusqu'à présent la méthode se base uniquement sur des fusions de demi-arêtes et les sommets de la reconstruction finale forment un sous-ensemble du nuage de points initial. Cela peut conduire à des résultats sous-optimaux surtout en présence de bruit ou de points aberrants. La reconstruction est donc couplée à un processus de déplacement des sommets de $\mathcal{C}$.

Après la fusion d'une demi-arête e, le sommet résultant v est déplacé en itérant deux étapes: (1) Pour un plan de transport π donné, v est déplacé vers la position qui minimise le coût de transport; (2) Les positions des sommets sont gelées et le plan de transport π est mis à jour de manière à minimiser le coût.

Algorithm 3: Algorithme de décimation.

Entrée: Complexe simplicial $\mathcal{C}$, nuage de points $\mathcal{S}$, nombre de sommets cible V
Sortie : Complexe simplicial $\mathcal{C}_{\text{final}}$ avec V sommets
for *chaque arête* $e \in \mathcal{C}$ **do**
  Simuler les opérateurs de fusion des demi-arêtes;
  Ajouter ces demi-arêtes à la queue de priorité $\mathcal{P}$ triée par changement de coût de transport δ induit par la fusion;
repeat
  Extraire la première demi-arête e^* de $\mathcal{P}$;
  Rassembler $\mathcal{E}$ les arêtes dont le voisinage intersecte le voisinage de e^*;
  Fusionnere^* et mettre à jour le plan de transport dans le voisinage de e^*;
  Mettre à jour $\mathcal{P}$ en recalculant les changements de coût Δ pour toutes les arêtes de $\mathcal{E}$.
until $\mathcal{C}$ *a exactement* V *sommets*;

Dans la première étape, nous calculons la position de v localement optimale lorsque le plan de transport π est fixé (les valeurs m_{ij} pour tout i et j sont fixées). Les positions des centres des cellules sont exprimées en coordonnées barycentriques par rapport aux sommets du triangle auquel ils appartiennent. Pour trouver la position optimale de v du triangle $t = (v, v_1, v_2)$, on minimise:

$$\min_v \sum_i \sum_j m_{ij} \| p_i - \alpha_j v - \beta_j v_1 - \gamma_j v_2 \|^2,$$

où $\alpha_j, \beta_j, \gamma_j$ sont les coordonnées barycentriques du centre de la cellule b_j par rapport aux sommets (v, v_1, v_2). La position optimale par rapport au triangle t est:

$$v^\star(t) = \frac{\sum_i \sum_j m_{ij} \alpha_j (p_i - \beta_j v_1 - \gamma_j v_2)}{\sum_i m_{ij} \alpha_j^2}.$$

Ainsi, chaque triangle t adjacent à v contribue d'une position optimale $v^\star(t)$. De plus le sommet lui même peut avoir des points initiaux qui lui sont assignées, il contribue donc également à son déplacement:

$$v^\star(v) = \frac{\sum_i m_{ij} p_i}{\sum_i m_{ij}}.$$

où m_{ij} est la portion de masse de p_i assignée à la cellule b_j du sommet v. Ainsi chaque simplexe (sommet ou facette) adjacent à v donne une position optimale pour v et la position

finale $v^\star$ est égale à la moyenne de ces positions optimales pondérées par la masse du simplexe correspondant.

$$v^\star = \frac{m(v)\cdot v^\star(v) + \sum_{t \text{adjacents à} v} m(t)\cdot v^\star(t)}{m(v) + \sum_{t \text{adjacents à} v} m(t)},$$

où $m(t)$ est la masse totale transportée sur le simplexe t (correspondant à la variable l_i dans la formulation LP avec i l'index du simplexe t). v est finalement déplacé à mi-chemin entre sa position courante et sa position optimale $v^\star$.

La deuxième étape gèle les positions des sommets et met à jour le plan de transport π par relaxations locales (algorithme 2). En alternant ces deux étapes, les sommets sont déplacés vers leur position optimale localement ce qui permet de mieux préserver les arêtes et les bords de la surface. La figure 5 montre un exemple de déplacement en 2D.

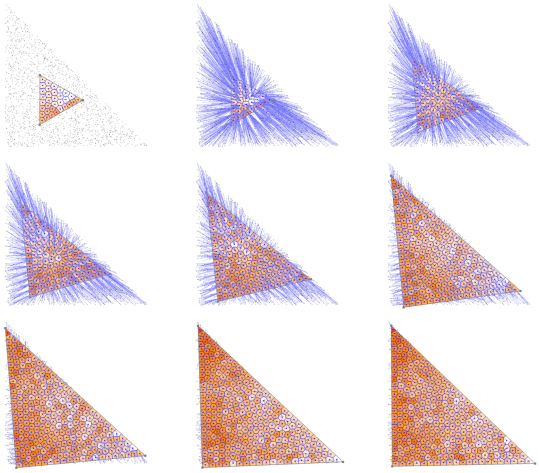

Figure 5: *Déplacement des sommets. Par soucis de clarté, on donne un exemple 2D d'un seul triangle avec seulement des cellules sur les facettes. Nous montrons d'abord le nuage de points initial, le complexe formé d'une seule facette et les cellules avec leurs capacités. Les images suivantes montrent les itérations du déplacement avec le plan de transport (en bleu).*

4.4. Filtrage des facettes

A la fin de la décimation, le complexe simplicial contient des facettes avec des mesures très faibles mais néanmoins non nulles dans le cas général, à cause du bruit et des points aberrants. Ces facettes sont retirées en filtrant les facettes par leur densité (le rapport entre leur mesure et leur aire). La figure 6 montre les effets d'un tel filtrage.

4.5. Résultats expérimentaux

L'algorithme a été implanté en C++ en utilisant la bibliothèque CGAL ([CGA]) pour initialiser la reconstruction. La

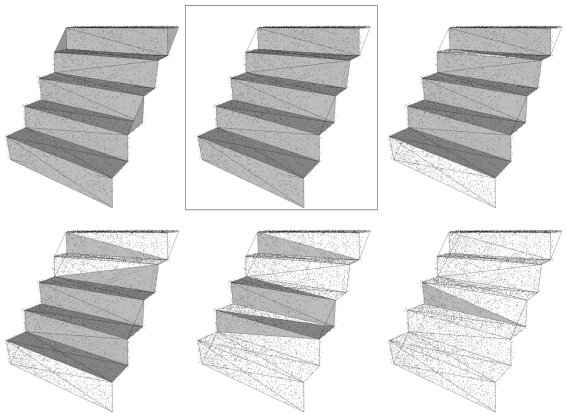

Figure 6: *Filtrage des facettes. Pour un nuage d'entrée bruité, le complexe simplicial final contient des facettes de faible mesure (en haut à gauche). Les autres images montrent le résultat du filtrage quand le seuil augmente. Le meilleure reconstruction est encadrée.*

bibliothèque Coin-Or Clp ([CLP]) a été utilisée pour résoudre le programme linéaire. L'implantation est partiellement parallélisée pour accélérer les calculs, en exploitant le fait que les simulations de fusion de demi-arêtes sont indépendantes.

L'initialisation et mises à jour de la queue de priorité sont de loin les opérations les plus coûteuses, car chaque fusion implique en moyenne 120 simulations. Avec une queue de priorité exhaustive sur un ordinateur à deux coeurs, la reconstruction d'un nuage de 30000 points prend une dizaine d'heures (de 3000 sommets pour le complexe initial jusqu'à 200 sommets pour le complexe final). Sur un serveur de calcul de 8 coeurs, ce calcul se réduit à 2h (grâce à la parallélisation et à des processeurs plus rapides). En utilisant une stratégie à choix multiples les temps sont environ 3 fois plus courts. Près de 70% du temps est passé à résoudre des problèmes linéaires. L'utilisation mémoire reste néanmoins assez faible durant tout le processus.

Robustesse au bruit. La figure 7 montre la robustesse de l'algorithme au bruit. Un bruit uniforme synthétique est ajouté à une forme d'escalier. On constate que même en présence d'une quantité significative de bruit, la méthode tend à retrouver les arêtes vives de la forme initiale. Si l'amplitude du bruit est supérieure à 5% de la boîte englobante, la méthode échoue, les facettes parasites ne pouvant pas être supprimées par simple seuillage.

Robustesse aux points aberrants. La figure 8) montre le résultat de l'algorithme face à un nuage dégradé par la présence de points aberrants. Les résultats restent bons jusqu'à 15% de points abérrants, mais la méthode échoue quand le pourcentage de points aberrants dépasse 20%, ce qui reste acceptable pour les méthodes d'acquisition actuelles.

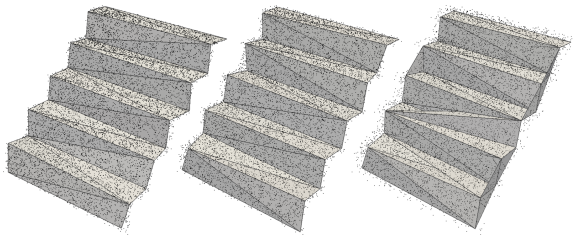

Figure 7: *Robustesse au bruit. La quantité de bruit ajouté est de* $\sigma = 1$ *(à gauche),* $\sigma = 2\%$ *(milieu) et* $\sigma = 5\%$ *(à droite), exprimée en pourcentage de la taille de la boite englobante. La reconstruction échoue à partir de* $\sigma = 5\%$.

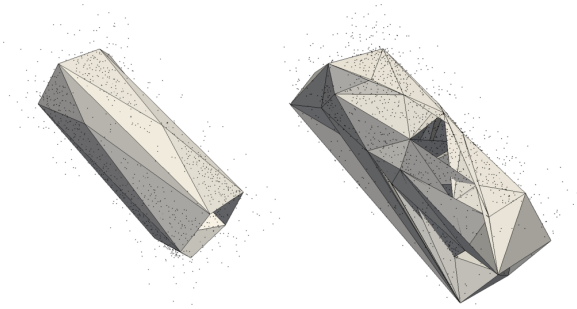

Figure 8: *Robustesse aux points aberrants. La reconstruction est efficace même avec* 10% *de points aberrants (à gauche) mais échoue à partir de* 20%. *Les points aberrants sont ajoutés aléatoirement avec une boîte englobante de* 120% *du nuage de points initial.*

Préservation des arêtes vives. La figure 9 montre cette préservation même avec des caractéristiques très fines que les méthodes implicites (ici Poisson) tendent à filtrer et à convertir en des défauts topologiques importants.

Sur le modèle du cône (figure 10), toutes les caractéristiques (sommet, bordures) sont préservées et la simplification est efficiente. De même sur un cylindre (fig. 11), les frontières sont préservées et la simplification conduit à des

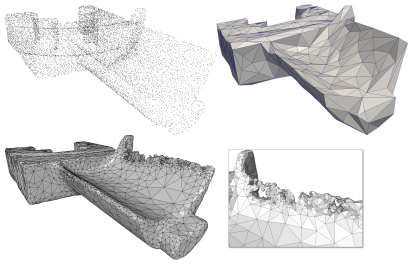

Figure 9: *Reconstruction du modèle de la lame contenant* 30*K points. Haut: la reconstruction finale. Bas: reconstruction de Poisson (avec raffinement de Delaunay pour le maillage d'une surface de niveau de la fonction implicite). Les détails montrent une arête vive avec un angle très faible où l'approche implicite échoue.*

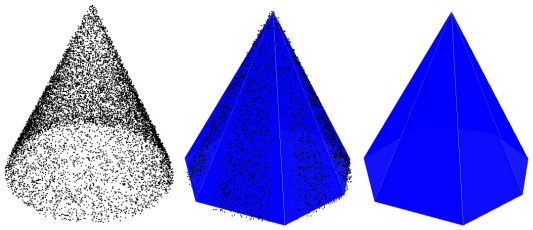

Figure 10: *Reconstruction d'un cône. A gauche: nuage de points initial. Milieu: nuage de points et complexe simplicial. Droite: reconstruction finale.*

triangles anisotropes dont la plupart des arêtes sont alignées avec la direction de courbure minimale.

La figure 12 illustre le comportement de cet algorithme sur deux polygones planaires qui s'intersectent. Jusqu'à 10 sommets, le complexe simplicial maintient la topologie, cependant passer de 10 à 8 sommets (le nombre attendu de sommets) demanderait un ensemble d'opérateurs topologiques plus riche pour déconnecter les deux polygones.

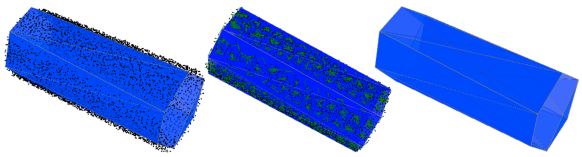

Figure 11: *Reconstruction et simplification d'un cylindre. Gauche:* 10*K échantillons bruités et reconstruction à 12 sommets (les densités par facette sont montrées). Milieu: Plan de transport. Droite: complexe simplicial et densité par facette.*

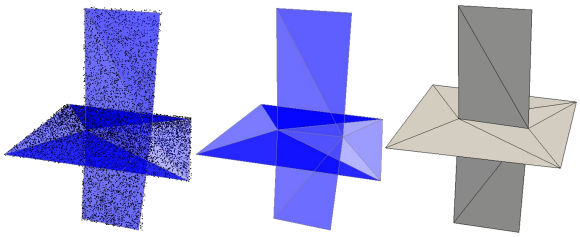

Figure 12: *Reconstruction et simplification de deux polygones planaires s'intersectant jusqu'à obtenir 10 sommets.*

La figure 13 montre les performances de la méthode sur des nuages de points LIDAR. Même avec ces données bruitées, notre méthode retrouves les arêtes vives de la forme et produit un maillage de faible complexité.

Limitations de la méthode. Les solveurs de programmes linéaires actuels sont très coûteux en temps de calcul, ce qui rend cette méthode trés lente et inefficace pour les grands nuages de points. De plus, rien ne favorise une 2-variété dans

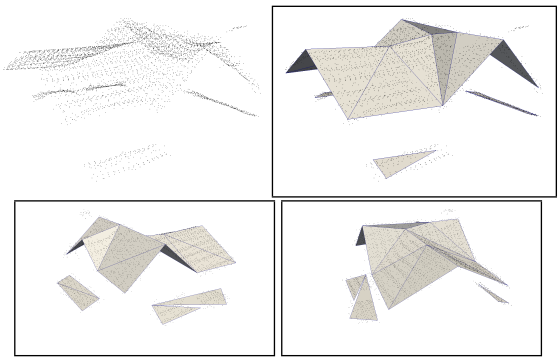

Figure 13: *Reconstruction d'un nuage de points LIDAR aérien représentant le toît d'une maison. En haut: nuage de points initial, milieu: reconstruction finale, bas: deux autres vues. La reconstruction donne une figure très simplifiée malgré le bruit (donnée de Qian-Yi Zhou et Ulrich Neumann).*

notre formulation, ce qui pose des problèmes de couverture multiples (figure 14).

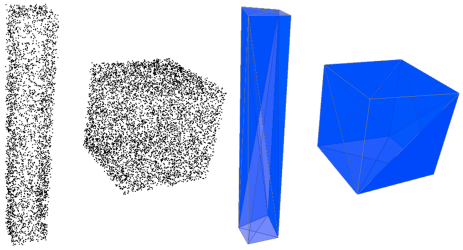

Figure 14: *Reconstruction et simplification d'une scène composée de deux boîtes. Gauche:* $10K$ *points bruités. Droite: reconstruction jusqu'à* 16 *sommets. Le niveau d'anisotropie est celui que l'on attend, mais certaines facettes sont couvertes deux fois.*

5. Post-traitement: récupération d'arêtes vives

Une autre application de la métrique proposée est de récupérer les arêtes vives et bords à partir du résultat d'une méthode de reconstruction (par exemple [KBH06]) qui produit une surface lisse et fermée. Ces méthodes passent bien à l'échelle mais lissent les aspérités et ferment les trous. Ceci peut être amélioré en appliquant un déplacement des sommets guidée par le transport et filtrage des simplexes.

Ce post-traitement prend en entrée une surface triangulée (la sortie d'une méthode reconstruction lisse comme la reconstruction de Poisson) et le nuage de points ayant conduit à cette reconstruction. Des cellules sont échantillonnées sur le maillage. Le plan de transport est ensuite calculé par la méthode de relaxation locale (section 3.3): chaque échantillon est assigné au plus proche vertex et les nouveaux plans de transport locaux sont itérés jusqu'à ce qu'un minimum local du coût de transport soit atteint. Chaque sommet du

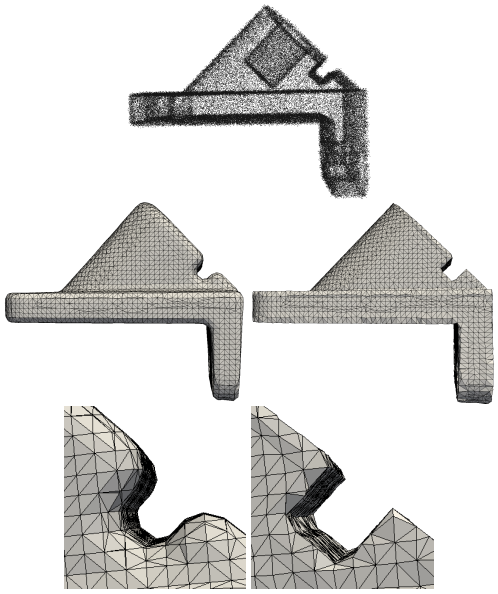

Figure 15: *Une ancre. Nuage de points bruité (en haut), reconstruction de Poisson (milieu gauche), reconstruction améliorée (milieu droite) et détails.*

maillage est ensuite déplacé (voir section 4.3) en calculant la direction de déplacement, en déplaçant le sommet dans cette direction et en mettant à jour le plan de transport. Ce processus dépend de l'ordre de parcours des sommets. Les figures 15, 16 et 17 montrent comment ce traitement permet de retrouver les arêtes vives.

Pour les surfaces comportant des trous et des bords, cette méthode permet de retrouver ces bords en combinant le déplacement de sommets et un filtrage qui supprime les facettes de mesure nulle. C'est assez clair pour l'église (figures 18 et 19). Sur cet exemple ($23K$ sommets et $232K$ points dans le nuage initial) il faut environ 10 minutes pour appliquer ce post-traitement.

6. Conclusion

Nous avons introduit une méthode de reconstruction de surface qui est à la fois robuste au bruit et aux points aberrants. Notre approche est basée sur la décimation d'un complexe simplicial guidé par une métrique de transport optimal entre la surface reconstruite et le nuage de points initial. Cette métrique est également utile comme outil de post-traitement pour les méthodes de reconstruction lisse. Le principal problème vient de son aspect calculatoire. Malgré nos stratégies de relaxation locale et de parallélisation, les nuages de points ne peuvent être reconstruits en un temps raisonnable. Cependant l'application de post-traitement montre tout l'intérêt de cette nouvelle métrique.

Remerciements. European Research Council (Starting Grant "Robust Geometry Processing", agreement 257474).

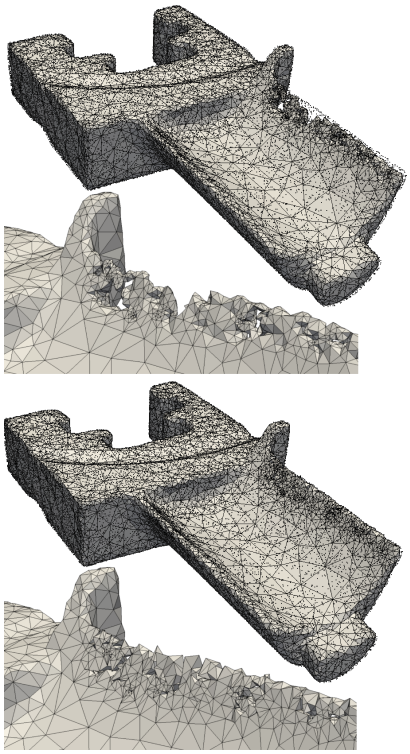

Figure 16: Blade. *Reconstruction de Poisson (2 premières lignes) et reconstruction améliorée (2 dernières lignes). Le nuage de points initial est représenté par des points noirs sur les vues globales et sont omis (par soucis de clarté) sur les détails. Les trous et artefacts visibles sur la figure 9 ne sont pas réparés mais les triangles sont attirés vers le nuage de points.*

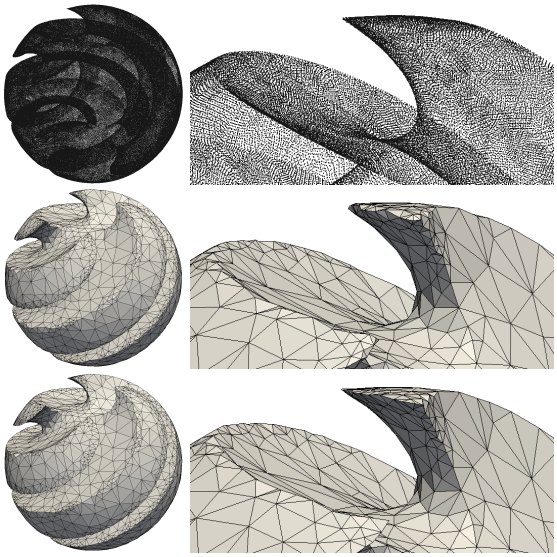

Figure 17: *Sphère. De haut en bas: nuage de points, reconstruction lisse (détail), et déplacement de sommets (détail).*

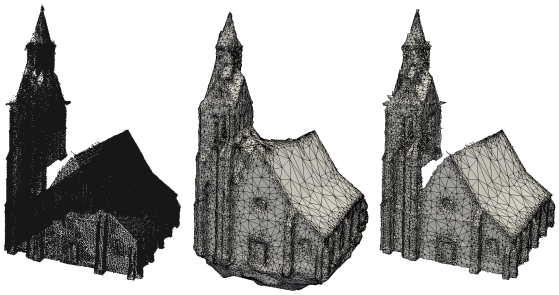

Figure 18: *Eglise. Nuage de points (à gauche), reconstruction de Poisson (au milieu). et maillage avec déplacement des sommets (à droite).*

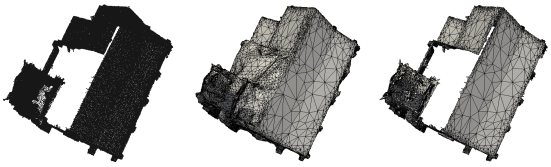

Figure 19: *Eglise. Nuage de points, Reconstruction de Poisson et amélioration par déplacement de sommets: un filtrage combiné au déplacement permet de retrouver les bords de la surface.*

Références

[AA06] ADAMSON A., ALEXA M.: Anisotropic point set surfaces. In *Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (2006), p. 13.

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *Eurographics Symp. on Geometry Processing* (2007), pp. 39–48.

[Ame99] AMENTA N.: The Crust algorithm for 3d surface reconstruction. In *Symposium on Computational geometry* (1999), pp. 423–424.

[ASGCO10] AVRON H., SHARF A., GREIF C., COHEN-OR D.: ℓ_1-sparse reconstruction of sharp point set surfaces. *ACM Trans. on Graphics. Vol. 29*, Num. 5 (2010), 1–12.

[BC01] BOISSONNAT J.-D., CAZALS F.: Coarse-to-fine surface simplification with geometric guarantees. *Computer Graphics Forum. Vol. 20*, Num. 3 (2001), 490–499.

[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using Lagrangian mass transport. *ACM Transactions on Graphics (SIGGRAPH Asia)* (2011).

[CGA] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[CLP] CLP, coin-or linear program solver. http://www.coin-or.org/Clp/.

[DCSA*13] DIGNE J., COHEN-STEINER D., ALLIEZ P., GOES F., DESBRUN M.: Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of Mathematical Imaging and Vision* (2013), 1–14.

[dGCSAD11] DE GOES F., COHEN-STEINER D., ALLIEZ P., DESBRUN M.: An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum. Vol. 30*, Num. 5 (2011), 1593–1602.

[DMSL11] DIGNE J., MOREL J.-M., SOUZANI C.-M., LARTIGUE C.: Scale space meshing of raw data point sets. *Computer Graphics Forum. Vol. 30*, Num. 6 (2011), 1630–1642.

[DTS01] DINH H. Q., TURK G., SLABAUGH G.: Reconstructing surfaces using anisotropic basis functions. In *International Conference on Computer Vision* (2001), pp. 606–613.

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C.: Robust moving least-squares fitting with sharp features. In *ACM SIGGRAPH 2005 Papers* (2005), p. 552.

[GWM01] GUMHOLD S., WANG X., MACLEOD R.: Feature extraction from point clouds. In *International Meshing Roundtable* (octobre 2001), pp. 293–305.

[HK06] HORNUNG A., KOBBELT L.: Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information. In *Eurographics Symp. on Geometry Processing* (2006), pp. 41–50.

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics. Vol. 28*, Num. 5 (2009).

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Eurographics Symp. on Geometry Processing* (2006), SGP '06, pp. 61–70.

[KSO04] KOLLURI R., SHEWCHUK J. R., O'BRIEN J. F.: Spectral surface reconstruction from noisy point clouds. In *Eurographics Symp. on Geometry Processing* (2004), pp. 11–21.

[LCOL07] LIPMAN Y., COHEN-OR D., LEVIN D.: Data-dependent MLS for faithful surface approximation. In *Eurographics Symp. on Geometry Processing* (2007), p. 67.

[LCOLTE07] LIPMAN Y., COHEN-OR D., LEVIN D., TAL-EZER H.: Parameterization free projection for geometry reconstruction. *ACM Transactions on Graphics. Vol. 26*, Num. 3 (2007), 22.

[LD10] LIPMAN Y., DAUBECHIES I.: Surface comparison with mass transportation. ArXiv preprint 0912.3488, 2010.

[LPK09] LABATUT P., PONS J.-P., KERIVEN R.: Robust and efficient surface reconstruction from range data. *Computer Graphics Forum. Vol. 28*, Num. 8 (2009), 2275–2290.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. In *ACM SIGGRAPH* (2003), vol. 22(3), pp. 463–470.

[OGG09] OZTIRELI C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum* (2009), vol. 28(2), pp. 493–501.

[PFR11] PEYRÉ G., FADILI J., RABIN J.: *Wasserstein Active Contours*. Tech. rep., Preprint Hal-00593424, 2011.

[PKG03] PAULY M., KEISER R., GROSS M.: Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum. Vol. 22*, Num. 3 (septembre 2003), 281–289.

[PP09] PANG X.-F., PANG M.-Y.: An algorithm for extracting geometric features from point cloud. *International Conference on Information Management, Innovation Management and Industrial Engineering. Vol. 4* (2009), 78–83.

[RAGS01] REINHARD E., ASHIKHMIN M., GOOCH B., SHIRLEY P.: Color transfer between images. *IEEE Comput. Graph. Appl.. Vol. 21*, Num. 5 (2001), 34–41.

[RDG10] RABIN J., DELON J., GOUSSEAU Y.: Regularization of transportation maps for color and contrast transfer. In *IEEE International Conference on Image Processing* (sept. 2010), pp. 1933 –1936.

[RPC10] RABIN J., PEYRÉ G., COHEN L. D.: Geodesic shape retrieval via optimal mass transport. In *European Conference on Computer Vision: Part V* (Berlin, Heidelberg, 2010), ECCV'10, pp. 771–784.

[RTG00] RUBNER Y., TOMASI C., GUIBAS L. J.: The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision. Vol. 40*, Num. 2 (2000), 99–121.

[Vil10] VILLANI C.: *Topics in Optimal Transportation*. American Mathematical Society, 2010.

[WCS05] WALDER C., CHAPELLE O., SCHÖLKOPF B.: Implicit surface modelling as an eigenvalue problem. In *Machine Learning ICML 2005* (2005), pp. 936–939.