Analyse du mouvement d'un objet déformable décrit par une série de nuages de points

François Destelle

Rapport interne

Analyse du mouvement d'un objet déformable décrit par une série de nuages de points

François Destelle

25 janvier 2012

Table des matières

1 Introduction

Ce rapport technique a pour but de mettre en forme nos démarches de recherche concernant le sujet de post-doctorat qui m'a été proposé, celui-ci ayant trait au suivi (*tracking*) de formes en mouvement. Je mets ici en évidence, d'une part, les buts que nous souhaitons concrètement atteindre, et d'autre part, les principales contraintes techniques qui leurs sont liées. Ces contraintes ont un rapport direct avec les données que nous traitons : les nuages de points. J'exposerai une méthode développée dans le but de répondre à cette problématique, en la situant par rapport au travail existant en suivi de forme. Enfin, je proposerai un ensemble de discussions ayant trait à la poursuite de ce travail de recherche.

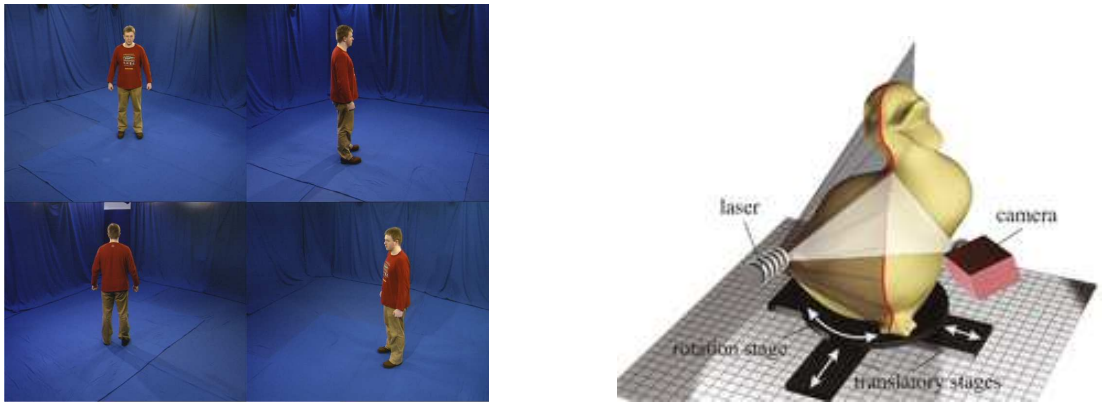
1.1 Contexte : sujet du post-doctorat

Le sujet initial de ce travail, proposé dans le cadre du projet $RAHA\ 3D$, a été défini de la façon suivante :

"Dans le cadre d'un projet de reconstruction tridimensionnelle à partir de séquences vidéo multicaméra pour l'analyse ou le suivi du mouvement de personnages ou plus généralement d'objets déformables en temps interactif et sans marqueur, nous nous proposons d'étudier le suivi des maillages construits à des instants successifs et d'étudier des déformations de ces derniers. À partir d'une séquence de maillages 3D obtenue par une phase de reconstruction, l'objectif est d'effectuer l'analyse et la mise en correspondance dynamique des maillages. Plusieurs méthodes de suivi des maillages peuvent être envisagées, reposant principalement sur des mesures de similarités, avec une représentation des objets incluant éventuellement des squelettes dynamiques et des hypothèses ou des contraintes sur le mouvement. L'utilisation de la multirésolution est

également possible. Une première partie du travail consistera à compléter un état de l'art des méthodes dans ce domaine. Une mise en oeuvre préliminaire d'une ou deux méthodes dégagées (un exemple possible : une méthode avec squelette et une sans squelette) sera effectuée pour mettre en évidence les verrous scientifiques encore à traiter. Des données seront tirées des sites publics ou proviendront de reconstructions dans le domaine de la rééducation fonctionnelle."

Au fil de nos discussions, ce sujet initial a été partiellement redéfini. Dans le cadre de ce projet, les données que nous possédons initialement sont obtenus par un scanner laser de la scène, que nous détaillons brièvement en partie suivante. Ces données sont constituées d'un ensemble de positions tridimensionnelles mesurées à la surface de l'objet en mouvement : il s'agit de nuages de points. À chaque phase de capture, que nous nommons *frame*, nous obtenons un nuage. Ainsi, ce travail de recherche considère l'analyse du mouvement d'un objet par rapport à une série temporelle de nuages de points.



(*a*) (*b*)

FIGURE 1 : *En (a) un exemple de scène filmée par plusieurs caméras disposées autour du sujet (image issue de [Star 05a]). En (b) nous illustrons le principe du scanner laser : la caméra ne mesure pas les données photométriques de la scène mais la distance d'un point de l'objet par rapport à l'émetteur du faisceau laser.*

1.2 Détails sur la capture de formes en mouvement

Afin de percevoir la profondeur d'une scène tridimensionnelle, une seule caméra (ou capteur) ne suffit pas. Nous parlons de vision stéréoscopique lorsque nous utilisons les mesures de deux caméras ; de façon plus générale, nous parlons de données multi-caméra lorsque nous étudions une scène au moyen d'un ensemble de caméras, voir Figure 1.a. Dans le cas de caméras photométriques standards, les données sont formées de plusieurs images relatives au point de vue de chacune d'elle. Un traitement annexe est alors nécessaire afin d'obtenir l'information de relief des objets de la scène. Le cadre de notre projet implique la capture de la scène au moyens d'émetteurs de rayon laser, voir Figure 1.b. Ce scanner laser nous permet d'obtenir directement un ensemble de mesure de profondeur, à la différence des caméras photométriques. Cependant cette méthode requière elle aussi un post-traitement, il nous faut synchroniser les différentes caméras laser afin d'obtenir une image de synthèse unique de la scène : il s'agit du recalage des données. Ces données de profondeur forment alors un ensemble de mesures ponctuelles, nous parlons alors de nuages de points.

De nombreux sites de recherches possèdent leur propre studio d'enregistrement multi-caméra, ils sont souvent très différents. Pour le cas de l'enregistrement photométrique aussi bien que par capture laser, citons le projet *Surfcap* [1], qui utilise 8 caméras haute définition et dont un jeu de données est mis à disposition. Le laboratoire du *MIT CSAIL* [2] met également à disposition ses enregistrements. Un autre projet est le *Civilian American and European Surface Anthropometry Resource Project* (*CAESAR*). Ils utilisent 4 scanners laser ; leur matériel est assez répandu dans la littérature, il s'agit de certains produits de la firme *Cyberware*.

1. *Centre for Vision, Speech and Signal Processing, University of Surrey, UK*, http ://www.ee.surrey.ac.uk/CVSSP/VMRG/surfcap.html
2. *MIT, Computer Science and Artificial Intelligence Laboratory,* http ://people.csail.mit.edu/drdaniel/mesh_animation/index.html

1.3 Objectifs

À long terme, ce projet a pour but de reconstruire une séquence animée des mouvements d'un patient à partir des captures laser sans marqueurs. L'idéal étant de pouvoir étudier n'importe quel objet en mouvement. Cette animation permettra de mettre éventuellement en évidence certains comportements moteurs déficients, ou peu souhaitables selon l'avis d'un spécialiste. Cette animation virtuelle pourrait se rendre utile de deux façons. En premier lieu, à la différence d'une capture vidéo classique, l'utilisateur pourrait déplacer son point de vue à n'importe quel endroit de la scène 3D, et ainsi concentrer son attention sur une zone précise du corps en mouvement. D'autre part, l'outil de reconstruction pourrait reproduire ce mouvement au ralenti, afin de mieux appréhender ses particularités. Dans le cadre du sujet d'étude présenté ici, nous nous concentrerons sur l'analyse des mouvements de la forme : le recalage des données multi-caméra et la reconstruction d'un maillage à partir des captures laser font l'objet d'autres travaux de recherche [3].

Ainsi, nous souhaitons construire la description d'un mouvement perçu à travers une série temporelle de nuages de points. Comme exposé précédemment, une application directe de cette description concerne l'interpolation des frames temporelles. En effet, si nous sommes en mesure d'interpoler (de préférence continuement) ces différentes frames, nous pouvons sur-échantillonner les données liées au mouvement, et ainsi lisser notre animation. Ce qui a pour conséquence directe de nous permettre d'améliorer la qualité de celle-ci, ainsi que d'en effectuer un ralenti. De plus, la caractérisation de ce mouvement est vraisemblablement en mesure d'apporter une information utile à d'autres techniques pouvant être appliquées par la suite sur les données capturées : la reconstruction d'un maillage ou la synthèse d'un squelette par exemple. Ceci car nous avons alors une relation de cohérence entre deux frames successives, cohérence que nous pouvons exploiter afin de guider ces traitements à posteriori.

1.4 Problématique

Le problème majeur auquel nous faisons face est lié à la capture des données. En effet, un ensemble de positions tridimensionnelles ne décrit pas explicitement une surface. Nous ne possédons pas d'information liée à la topologie de l'objet, puisque les points de notre nuage n'ont pas de relation directe les uns avec les autres. Nous considérons que ces informations de position dans l'espace constituent notre seule information, ce qui diffère d'un certain nombre de travaux actuels centrés sur la même problématique, que nous exposerons en section suivante. Toute la difficulté sera ainsi de mettre en correspondance deux (ou plus) ensembles de points désordonnés. De plus, nous choisissons dans un premier temps de ne pas ajouter d'information a priori sur l'objet en mouvement que nous analysons, dans le cas d'un être humain il s'agirait par exemple d'un squelette de référence.

3. À noter que la reconstruction d'un maillage n'est pas obligatoire pour ce qui est de la visualisation d'une scène 3D, De nombreuses méthodes permettent le rendu de nuages de points de façon très satisfaisantes [Rusi 00, Lins 01, Alex 03].

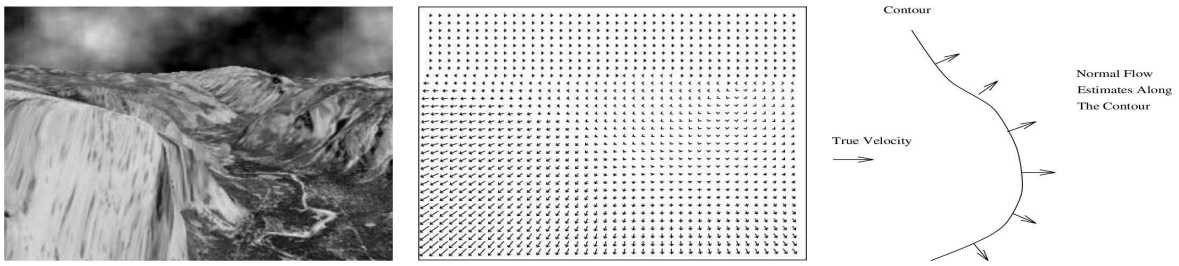
La qualité de cette capture est d'autre part toute relative, nous nous attendons à des zones cachées de l'objet que les caméras laser ne peuvent atteindre, ainsi qu'à des données légèrement bruitées. Par contre, nous admettrons que les mouvements décrits par la série temporelle de nuages sont de *faible* amplitude, ceci étant lié à la fréquence standard des caméras laser actuelles : de l'ordre de 15 à 30Hz. De plus, une capture par laser génère un ensemble de points relativement dense.

2 Un état de l'art du suivi de forme en mouvement

Les travaux actuels en suivi de formes en mouvement sans marqueurs sont principalement axés sur deux techniques : le suivi de maillages et le suivi de nuages de point. Ces deux approches sont très différentes, elles ne posent pas les mêmes problèmes. Nous pouvons néanmoins trouver certaines techniques communes au suivi de ces deux types d'objets, notamment les méthodes basées sur la paramétrisation des données, la construction d'un squelette ou celles liés aux champs de mouvement. La principale différence que nous pouvons noter entre tous ces travaux sur le suivi d'objet consiste en l'information que nous possédons au départ, c'est à dire celle que nous avons capturée de l'objet. De nombreux travaux sont en particulier basés sur une information de couleur mesurée sur l'objet, qu'il est impossible de faire avec des caméras laser.
Nous présentons ici un court état de l'art en considérant la séparation des techniques de suivi basées sur les maillages de celles qui ne possèdent pas d'information de topologie (les nuages de points). La première partie contiendra également un court exposé des différentes grandes familles de techniques utilisées. Par ailleurs, concernant l'étude de données volumiques (nous nous concentrons ici sur des relevés surfaciques), le lecteur pourra se référer par exemple à [Tree 01, Preu 03] et à l'état de l'art un peu moins récent [Laza 98].

2.1 Le suivi de maillages

Récemment, certains travaux de suivi de maillages ont montré d'excellents résultats, voir par exemple [Vara 08, Fran 10, Cagn 10]. Un maillage, contrairement à un nuage de points épars, possède une topologie explicite. En général nous parlons de maillages surfaciques, ils décrivent une 2-variété discrète. Cela constitue un apport d'informations important au nuage de points, car un maillage est une *surface discrète* qui est l'approximation de la surface de l'objet sous-jacent. Ceci permet d'évaluer directement des critères de forme tels que la notion de voisinage des sommets, les normales à la surface, ou encore la courbure des données en un point ; ce qui est bien plus difficile à déterminer sur un nuage de points. Par contre, les techniques associées au suivi de maillages en mouvement possèdent leur propre limitation, qui est le respect de la cohérence de cette topologie au cours du mouvement. En effet, le changement de topologie d'un maillage implique des suppressions et des ajouts de faces, ce qui est un procédé simple en lui même mais qui implique de lourds traitements afin de pouvoir valider la cohérence globale de cette topologie. De plus ces changements de topologie sont difficiles à détecter. Une séquence temporelle de maillages est dite *dynamique* si les différentes frames possèdent le même nombre de sommets et la même connectivité ; elle est dite *non contrainte* si ça n'est pas le cas [Arci 10a]. Nous nous concentrerons sur le deuxième cas, en rapport avec le cadre de notre analyse où les différents relevés de données ne possèdent pas le même nombre de sommets.



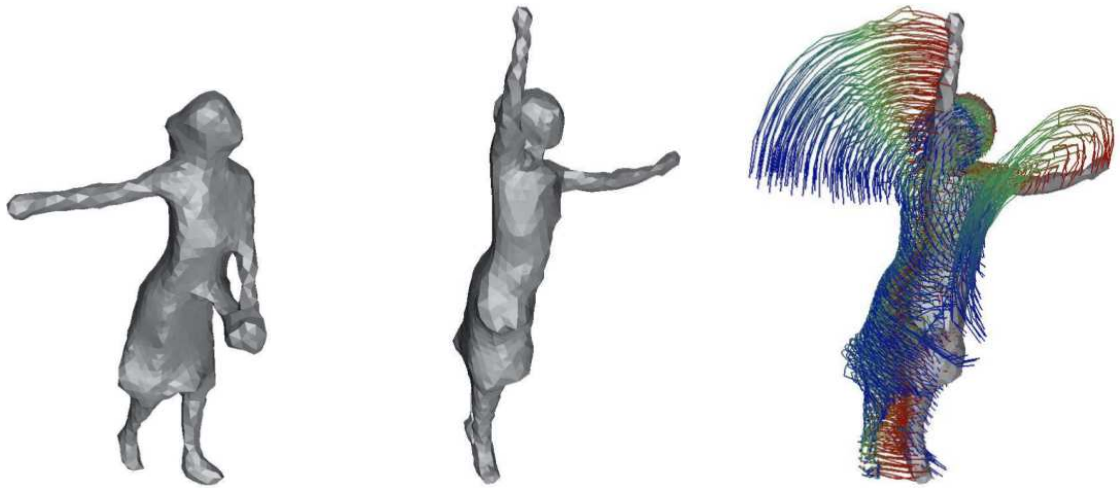
(a) (b) (c)

FIGURE 2 : *Exemples de flots optiques : (a) Une frame issue de l'image en mouvement* Yosemite*, nous modélisons en (b) ce mouvement par un flot optique. En (c) nous illustrons un flot optique appliqué aux normales d'un contour actif (images issues de [Beau 95]).*

Le problème de la cohérence topologique concerne les nombreuses méthodes de suivi de maillages basées sur la déformation d'un modèle de référence, généralement proche d'un corps humain ou d'un visage, et qui pour beaucoup utilisent un squelette, voir notamment les travaux [Deca 00, Carr 03, Agui 04, Salz 07, Cora 06]. À noter que le travail [Cora 06] correspond à l'approche médicale de notre projet. Certains travaux récents [Zaha 11] se sont néanmoins penchés sur la résolution efficace et robuste du changement de topologie d'un maillage en mouvement. Dans [Cagn 10], les auteurs utilisent eux aussi une méthode basée sur un modèle 3D de référence, mais ils considèrent celui-ci comme étant la première frame de leur série temporelle, qui peut décrire une scène complexe. Leur but est de déformer cette référence de frame en frame. Ils considèrent alors un ensemble de zones géométriques locales sur le maillage, les *patchs géodésiques* (cette technique est donc difficilement applicable à un nuage de points). Ils formalisent ensuite le problème en considérant une estimation Bayésienne du maximum de vraisemblance de mouvement entre ces patchs, qu'ils résolvent par un algorithme EM (*Expectation-Maximization*[Demp 77]). Cette méthode présente de très bons résultats [4]. Elle est considérée comme robuste aux données bruitées et est applicable à n'importe quel type de scène 3D. Par contre, elle ne prend pas en compte les changements de topologie du maillage de référence, et étant relativement complexe à mettre en œuvre, elle n'est pas temps réel (de l'ordre de 20 à 30 secondes de calcul entre deux frames contenant relativement peu de sommets).

Une autre grande famille de techniques de suivi repose sur le *flot de scène*, ou flot *de déplacement*. Nous retrouvons également l'application de cette approche pour le suivi de nuages de points en section suivante. Le principe du flot de scène 3D repose sur le flot optique, utilisé en traitement d'images classique [Beau 95], voir Figure 2.a. Pour les suivi de formes en 2D, les contours actifs par exemple se basent bien souvent sur le flot de déplacement de leurs normales (Figure 2.b). Le mouvement est alors conceptualisé au moyen d'un champ de vecteurs de déplacement. Le flot de scène 3D est un flot optique défini en tout point d'un espace tridimensionnel [Vedu 99]. Celui-ci peut être discret (Figure 2.b), donc défini sur un octree [Vedu 00], il peut être défini comme une surface implicite [Keri 06] ou encore un maillage [Neum 02, Agui 07]. À noter que cette technique par flot de scène à été utilisée pour le *morphing* (ou *blending*) de forme. Il s'agit d'une problématique proche de la notre et nous y reviendrons en partie 4. Un bon récapitulatif sur les différentes techniques de morphing est présenté dans [Alex 02].

4. La vidéo de démonstration est disponible ici :
http ://campar.cs.tum.edu/files/publications/cagniart2010ECCV.video.avi



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

FIGURE 3 : *Illustration de la technique développée par [Vara 08]. En (a) le maillage issu de de la première frame, on souhaite le faire évoluer à terme vers (b) (il ne s'agit pas de la seconde frame mais de la frame finale de l'animation). En (c) nous présentons le flot de scène que les auteurs construisent.*

Une autre catégorie de techniques utilisée en suivi de forme 3D est basée sur la détection de zones d'intérêt particulier sur les maillages [Bron 07a, Star 07, Arci 10b]. Leur principe réside dans l'identification de zone géométriques ou photométriques bien reconnaissables sur la forme (la saillance par exemple), qui permettent par diffusion un suivi global de l'objet en mouvement.
Ces techniques sont d'ailleurs généralement appliquées en reconnaissance de forme (*shape matching*, voir un état de l'art récent [Tang 08]), et elles sont assez difficiles à adapter aux nuages de points. Les travaux [Vara 08] proposent une méthode intéressante de suivi de maillages basée sur cette approche. Leurs données sont constituées de maillages texturés, donc possédant une information de couleur en tout sommet du maillage. Leur système est exposé en trois phases : (1) extraction et mise en correspondance de données géométriques et photométriques éparses, (2) diffusion laplacienne de ces informations sur tout le maillage, et (3) morphing du premier maillage vers le second au moyen d'un flot de scène, voir Figure 3. Ce système présente des résultats également très convaincants, mais sa mise en œuvre est là encore rendue difficile de part l'ensemble des processus itératifs à appliquer, et les temps de calcul (non précisés) semblent très importants pour chaque mise en correspondance de maillage.

Enfin, certaines méthodes se basent sur la paramétrisation des maillages. Le domaine de paramétrisation est un repère commun aux deux maillages permettant leur mise en correspondance [Zhan 99, Zige 02, Star 05b] (voir [Xiao 04] concernant les nuages de points, illustré en Figure 4). La paramétrisation est, de façon grossière, un changement de repère des données qui permet de réduire leur dimension. Cette forme de paramétrisation commune peut être difficile à déterminer de façon satisfaisante ; certains travaux se sont alors penchés sur la paramétrisation directe de l'objet de départ dans l'espace de l'objet déformé [Bron 07b]. De façon générale, ces méthodes sont difficiles à mettre en place et, dans notre optique de recherche, ne possèdent pas de réel avantage sur les autres approches, en plus de se limiter à une topologie globale constante de l'objet en déformation.

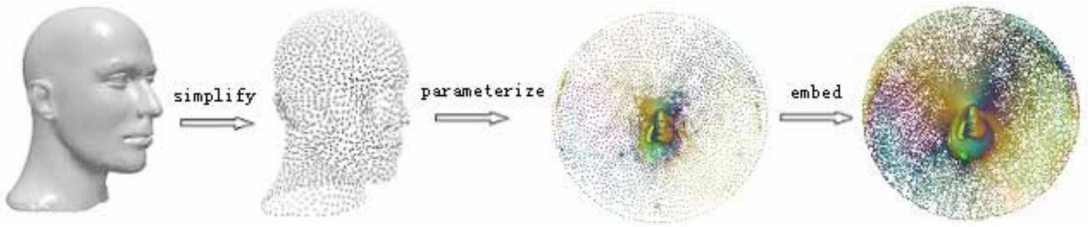


FIGURE 4 : *Illustration de la technique développée par [Xiao 04], basée sur la paramétrisation des données. Celles-ci sont tout d'abord simplifiées, puis l'espace de paramétrisation est déterminé, il s'agit ici d'un disque. À droite : paramétrisation des données originales avant leur simplification.*

2.2 Le suivi de nuages de points

L'utilisation de modèles 3D basés sur des points, sans information de connectivité, se développe de façon très rapide depuis quelques années. La prise en compte de points comme primitives de la modélisation de données possède l'avantage d'être plus légère que celle par maillage. Le lecteur pourra se référer aux ouvrages récapitulatifs de l'état de l'art en rendu, traitement et animation d'objets modélisés par points [Alex 04, Gros 07]. Comme dit précédemment, la difficulté importante du suivi de nuages de points réside dans le manque d'informations que nous possédons. Par rapport au suivi de maillage, nous ne nous inquiétons pas des changements de topologie, puisque la notion de 2-variété n'a plus de sens dans ce cadre. Concrètement, si nous souhaitons synthétiser une suite de maillages afin de les visualiser, nous déplaçons alors le problème : la reconstruction d'un maillage à partir du nuage est considérée comme une phase de post-traitement, elle ne dépend pas des techniques de suivi.

De façon générale, assez peu d'études se focalisent sur le suivi de nuages de points acquis par capture laser, la grande majorité des nombreux travaux existants se basent sur la capture par caméras vidéos. Certains travaux traitent de la mise en correspondance d'ensembles de points désordonnés, voir par exemple l'étude récente [Myro 10] qui traite surtout de reconnaissance de forme et de morphing en deux dimensions. Leur méthode est basée sur une approche probabiliste, ils ne mettent plus en correspondance deux nuages de points mais deux mélanges de probabilités de présence, en déterminant un maximum de vraisemblance. Cette technique est particulièrement robuste aux données bruitées. Elle présente également un état de l'art de la reconnaissance de forme basée sur des nuages de points.

La majorité des techniques en rapport avec les nuages de points utilisent une discrétisation des données, souvent sous la forme d'une structure de type *octree*. Les techniques exposées dans [Sund 03, Xie 08] sont basées sur la construction d'un squelette de l'objet. Le travail [Yoon 10] présente un récapitulatif très conséquent des avancées techniques concernant la squelettisation appliquée à la capture du mouvement sans marqueur. Les auteurs utilisent un système multi-caméra photométrique, et non laser. La base de leur technique est axée sur une voxélisation originale des silhouettes issues des différentes caméras. Ils effectuent ensuite une squelettisation discrète de leurs données, qu'ils segmentent afin d'identifier les zones en mouvement.
L'étude [Cheu 00] propose une approche de squelettisation à partir d'ellipsoïdes créées par regroupement de voxels. Cependant leur méthode est limitée et lourde en terme de calculs.

FIGURE 5 : *Morphing de deux nuages de points, un lion vers un dinosaure, au moyen de la méthode introduite dans [Cmol 03].*

Enfin, un nombre conséquent de travaux ont été développés sur la base du flot de scène [Vedu 99], à savoir la description du mouvement de l'objet par un champ de vecteurs discret. Cette méthode est très utilisée notamment dans le domaine médical, pour l'analyse du mouvement de forme tridimensionnelles volumiques acquises par *IRM* (voir par exemple [Sieb 07]). Cependant ces méthodes profitent d'informations supplémentaires sur les données volumiques, telles que la couleur, car elles considèrent ce volume comme une pile d'images 2D dont chaque pixel représente une densité de matière.
Dans notre cadre, citons la méthode de morphing introduite par [Cmol 03], qui traite du morphing de nuages de points sans a priori sur la forme, ni ne possédant d'autres informations que la position de ces points. Nous proposons ici de la décrire car il s'agit de celle qui s'approche le plus de la technique que nous introduisons en partie suivante. Celle-ci repose sur quatre phases :

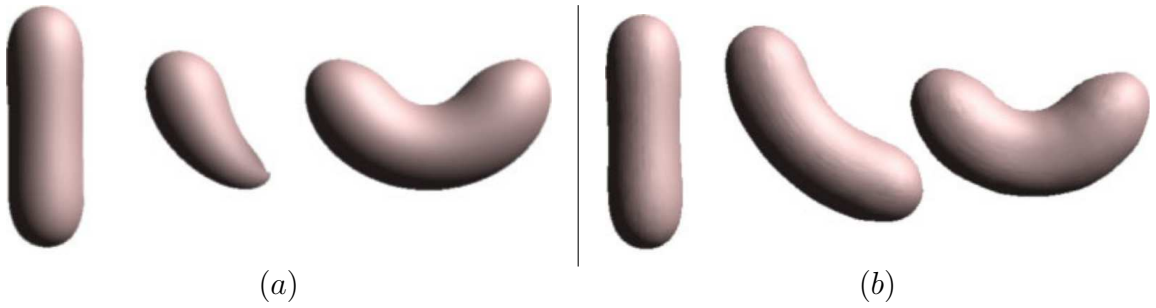
(1) Voxélisation adaptative des données constituées par les deux nuages de points au moyen d'un arbre binaire de partitionnement (*BSP*), en considérant les trois axes d'inertie principaux des points du voxel d'intérêt. L'arrêt de la subdivision est lié aux nombre de sommets présent dans chacun d'eux.

(2) Mise en correspondance des voxels contenant des points du nuage de la première frame $\mathcal{F}^1$ vers ceux du nuage de la frame suivante $\mathcal{F}^2$. Initialement, les deux voxels du premier niveau de subdivision u_0 et v_0 sont appariés. L'algorithme considère ensuite l'appariement des 8 voxels u_1 issus de la subdivision de u_0 avec les 8 voxels v_1 issus de v_0, et le procédé est récursif. L'appariement des voxels u_1 et v_1 est effectué en minimisant une mesure de distance entre leur centre de masses. Le procédé est itéré pour l'ensemble de l'arbre, générant ainsi un appariement complet de $\mathcal{F}^1$ vers $\mathcal{F}^2$.

(3) Mise en correspondance des points, cette fois, contenus dans chaque voxel de $\mathcal{F}^1$ vers les points contenus dans le voxel de $\mathcal{F}^2$ qui leur est apparié.

(4) Morphing des points par interpolation linéaire.

Une limitation importante de cette méthode réside dans l'appariement des voxels de l'arbre, à savoir des zones du nuage d'entrée vers les zones du nuage déformé. Ce procédé est entièrement basé sur la structure hiérarchique des deux arbres de partitionnement : si u_s et v_s sont appariés, les voxels u_{s+1} ne pourront être appariés autrement qu'avec les voxels v_{s+1}, ce qui n'a que peu de lien avec la forme réelle des deux nuages. De plus, leur métrique est limitée à une mesure de distance entre deux centres de masses. Or, cette phase de mise en correspondance est d'une importance centrale dans toutes les méthodes de suivi ou de morphing basées sur le flot de scène. De fait certains résultats de morphing utilisant cette méthode sont visuellement discutables, voir Figure 5.



(*a*) (*b*)

FIGURE 6 : *Illustration des travaux issus de [Bao 05]. En (a), la simple interpolation linéaire des points conduit à des résultats peu souhaitables. En (b), le résultat est amélioré par l'utilisation de contraintes physiques sur l'estimation de la surface.*

D'autre part, l'étude [Bao 05] se focalise, non plus sur l'appariement des zones de l'objet identifiées comme similaires entre deux frames, mais sur le chemin que devrait suivre une zone vers la zone déformée qui lui est associée. Ceci car lors d'un morphing, la simple interpolation linéaire d'un ensemble de points vers un autre peut conduire à de mauvais résultats, voir Figure 6. Leur méthode se base sur une méthode itérative de type *MLS*, proche de la méthode de simulation physique [Guo 05], minimisant un système de contraintes physique issues de l'énergie des plaques minces (*thin shell*). Dans [Xiao 04], les auteurs utilisent une interpolation linéaire comme chemin de déplacement des points. Ils constatent d'autres problèmes tels que l'apparition de zones où la densité de points devient très vite proche de zéro. Ceux-ci proposent de résoudre cette limitation par un sur-échantillonnage de leur résultat par la méthode exposée dans [Alex 03].

3 Une méthode temps-réel de suivi de nuages de points

Nous exposons ici notre méthode développée dans le but de suivre le mouvement de nuages de points. Celle-ci ne considère aucun a priori sur la nature de l'objet en mouvement (pas de squelette de référence, pas de topologie fixée), ni aucun a priori sur la nature de ces mouvements (rigides ou non). Notre parcours de recherche s'est axé sur le développement d'une méthode générale, robuste aux données bruitées, simple à mettre en œuvre et rapide en terme d'exécution. Nous remarquons également que notre système est indépendant de la dimension des données, les sommets du nuages peuvent appartenir à $\mathbb{R}^n, n \neq 3$. Cependant, de par le cadre de notre projet, nous nous sommes concentrés sur des mouvements de *faible* amplitude : nous effectuons un suivi de forme en mouvement, et non une méthode générale de morphing. Nous discuterons de cette limitation en partie 4.

Nous formulerons notre problème en partie suivante en y introduisant nos notations. Nous ferons un rapide exposé des arbres de partitionnement en partie 3.2, puis nous exposerons notre approche en partie 3.3. Enfin, nous décrirons notre système de suivi de mouvement à travers la création d'un flot de scène en partie 3.4.

Afin de simplifier l'exposé de notre méthode, nous prenons en compte le suivi de mouvement entre deux nuages de points particuliers : le nuage $\mathcal{N}_1$ vers le nuage $\mathcal{N}_2$.

3.1 Formulation du problème

Soit la série de nuages de points $\mathcal{N}_i, i \in [1, N]$. Ces nuages sont composés de n_i sommets de $\mathbb{R}^3$. Considérons maintenant deux nuages en particulier, par exemple $\mathcal{N}_1$ et $\mathcal{N}_2$. Les données $\mathcal{N}_1$ sont mesurées en un temps t_1 tandis que les données $\mathcal{N}_2$ sont mesurées en un temps t_2. Nous cherchons à estimer ce mouvement, ce qui revient à déterminer le chemin (ou *mapping*) allant de $\mathcal{N}_1$ vers $\mathcal{N}_2$ entre t_1 et t_2 :

$$m : \mathcal{N}_1 \rightarrow \mathcal{N}_2 \ .$$

Ceci se traduit par l'obtention des chemins des sommets $a_i \in \mathcal{N}_1$ vers les sommets $b_j \in \mathcal{N}_2$, $b_j = m_s(a_i)$, ces chemins forment le flot de scène. En réalité, étant donné que n_1 est potentiellement différent de n_2, nous définissons le mouvement par l'ensemble des couples de sommets $\mathcal{M}_s : \{(a_i, b_j) \mid a_i \in \mathcal{N}_1, b_j \in \mathcal{N}_2\}$.
L'obtention de cet appariement de sommets peut être fait de différentes manières. La plus simple est d'associer chaque sommet du premier nuage vers le sommet du deuxième nuage qui lui est le plus proche, ce qui reviendrait à minimiser

$$d_{min} = \sum_{(a_i,b_i)\in\mathcal{M}_s} ||a_i, b_i||^2 \ .$$

Cette méthode est impraticable pour deux raisons, l'une d'elle étant qu'il s'agit d'une minimisation très proche du problème bien connu du treillis (*lattice problem*) : trouver le vecteur de taille minimale dans un treillis est NP-Complet. Nous reviendrons sur la deuxième raison quand nous exposerons notre métrique (au sens d'une mesure de distance) en partie 3.4.1, il s'agit d'une question liée à l'orientation de la surface sous-jacente.

Une méthode classique de simplification du problème est l'utilisation d'arbres de partitionnement de l'espace. Ces structures multi-résolution permettent de découper ("*divide and conquer*") la contrainte initiale en plusieurs sous-problèmes plus simples à résoudre. En pratique, nous plongerons nos données (les nuages de points) dans des arbres de partitionnement, que nous subdiviserons en cellules plus petites. Puis nous chercherons l'appariement $\mathcal{M}_s$ localement à ces cellules (ou voxels) : au lieu de définir un chemin entre les points de deux nuages, nous définirons un chemin entre les cellules de deux arbres. Et c'est à partir de cet appariement de cellules que nous pourrons déterminer l'appariement des sommets d'un nuage à l'autre, et ainsi créer le flot de scène.

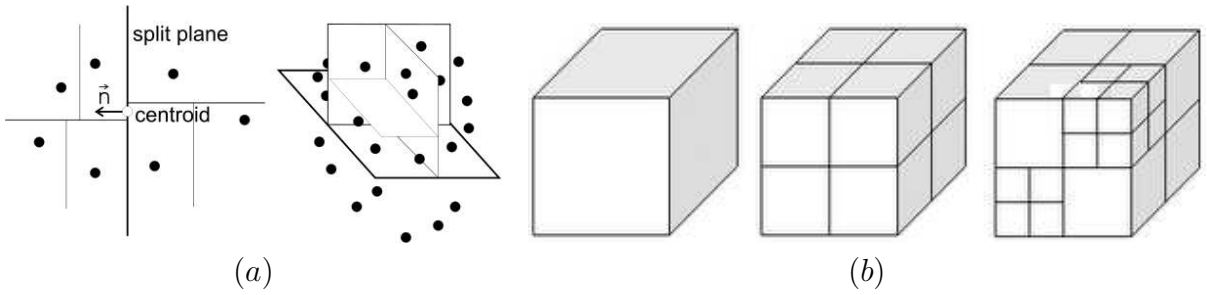
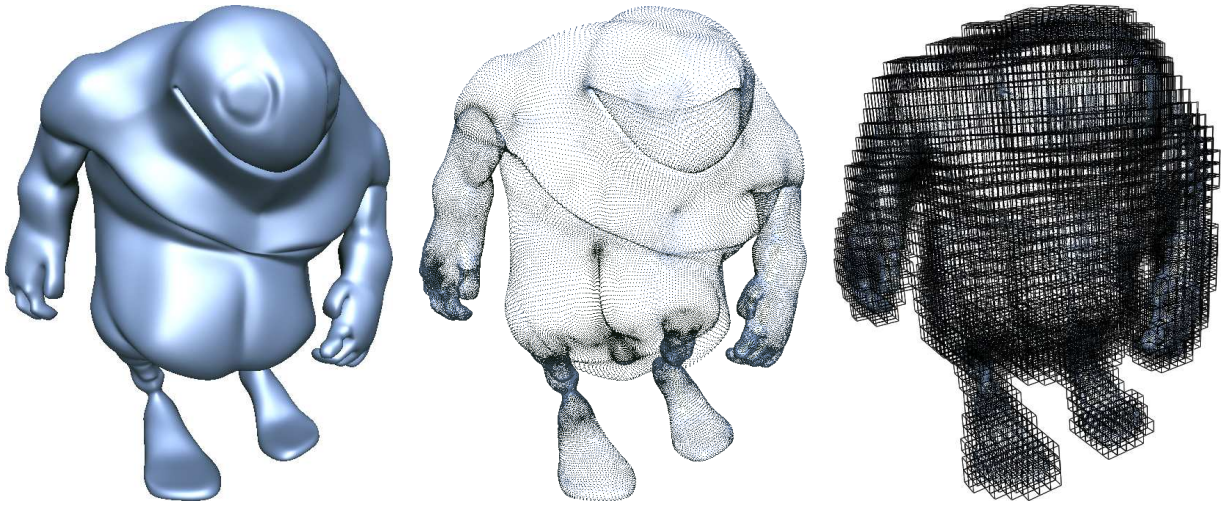


FIGURE 7 : *Illustration du principe de l'arbre de partitionnement binaire (a), en deux puis trois dimensions, et de l'octree régulier mais non uniforme (b), où seuls certains cubes sont subdivisés (en 8).*

3.2 Les arbres de partitionnement

Il existe de très nombreux types d'arbre de partitionnements de l'espace des données. Un arbre possède une cellule mère, ou racine, qui contient l'ensemble des données ; les arbres diffèrent par leur méthode de subdivision. Le principe de ces structures est de classer les données de façon récursive dans les cellules filles issues de la subdivision de la cellule racine. Les cellules finales, au niveau de subdivision maximal, sont les feuilles de l'arbre. Une fois l'arbre de partitionnement choisi, sa première cellule contient les données que nous souhaitons partitionner. Dans notre cas, nos données sont les points des nuages $\mathcal{N}_1$ et $\mathcal{N}_2$.

Parmi les structure les plus connues, la plus simple est le *BSP* (*Binary Space Partition*), ou *Btree*. Le principe est de subdiviser une cellule en deux, séparant ainsi les données entre deux cellules filles, voir Figure 7.a. En deux dimensions la subdivision est une droite (orientée par une normale), en trois il s'agit d'un plan ; cet arbre est donc binaire. Cette structure est très performante en terme de calculs et très simple à mettre en place. Le choix du plan de césure d'une cellule dépend de l'usage d'un tel arbre : césure régulière ou en fonction des axes d'inertie des données par exemple. Le *k-d* tree généralise ce concept binaire en dimension quelconque.
D'autres structures simples et très utilisées sont le quadtree et l'octree, voir Figure 7.b, respectivement en 2 et 3 dimensions. Leur principe repose sur la même idée mais cette fois il s'agit de carrés ou de cubes que l'on subdivise respectivement en 4 ou en 8. À ce propos, nous parlons de subdivision uniforme lorsque toutes les cellules sont subdivisées jusqu'au même niveau, et de subdivision régulière lorsque le sommet de scission (donc le centre des axes ou plans bissecteurs) est au centre de la cellule à subdiviser. L'utilisation de l'octree est très largement répandu à cet effet, notamment pour ce qui est de la reconstruction de nuages de points [Hopp 92, Boub 06] : l'idée générale de ces travaux est de reconstruire localement le nuage à l'intérieur de chaque voxel. Le principe est le même pour ce qui est du *marching cube* [Lore 87], utilisé entre autres pour la construction d'un maillage à partir d'une surface implicite.



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

FIGURE 8 : *Voxélisation d'une scène : (a) la surface (de synthèse) observée, (b) le nuage de points résultat d'un enregistrement laser et en (c) nous affichons les cellules de l'octree non vides (niveau de subdivision 5), elles contiennent des sommets du nuage. Lorsqu'une cellule est vide, elle n'est pas subdivisée au niveau suivant et nous ne la prenons plus en compte.*

L'approche présentée dans [Vedu 00] est basée sur le plongement de chaque nuage dans un octree. Leur méthode considère l'espace à 6 dimensions composé de ces deux structures. Celle-ci est cependant éloignée de notre problématique car elle dépend beaucoup de données photométriques que nous ne possédons pas, les techniques employées sont très proches du traitement d'image classique. Dans [Cmol 03], le choix des auteurs a été de plonger chaque nuage dans un *BSP*. Ils ont mis en place une subdivision basée sur le centre de masse des données, la scission des cellules étant orientée par leur axe d'inertie principal. Dans l'étude [Agui 04], axée sur les mouvements de l'être humain, les auteurs considèrent une approche similaire. Ils ne travaillent pas avec des nuages de points mais avec des voxels 3D, cependant ils considèrent le nuage de points formé par le centre de chaque voxel. Leur système plonge les points dans un arbre *BSP*, dont les cellules sont considérées comme des ellipsoïdes, en rapport avec les axes d'inertie des données présentes dans ces dernières. Suit une phase de regroupement des ellipsoïdes de chaque nuage basée sur un facteur de variance mesuré au voisinage de chaque voxel. Intuitivement, dans le cas d'un humain, les membres rigides tels que les bras ou les jambes auront tendance à être décrits par peu d'ellipsoïdes. Les auteurs apparient ensuite ces deux ensembles d'ellipsoïdes simplifiés. Notons que ces deux derniers travaux mettent en correspondance les cellules d'arbres potentiellement très différents.

3.3 Notre approche

Tout comme les travaux précédemment cités, nous optons pour l'utilisation d'un partitionnement spatial. Par contre, notre optique n'est pas seulement d'optimiser les traitements ni de grouper les zones du nuages : nous souhaitons travailler sur ces données par rapport à un repère discret 3D commun aux deux octrees.

Considérons les deux nuages $\mathcal{N}_1$ et $\mathcal{N}_2$, chacun étant plongé dans un octree à subdivision régulière. Un élément central de notre méthode est l'initialisation de la voxélisation : la cellule racine des deux octrees est la boite englobante de l'union des deux nuages. Nous subdivisions ensuite récursivement ces deux cellules, puis leurs cellules filles si celles-ci contiennent des données (la subdivision n'est donc pas uniforme), voir une illustration en Figure 9. Notre approche profite de plusieurs avantages importants de ce type d'arbre multi-résolution. Les points communs de notre méthode d'appariement avec l'étude présentée dans [Cmol 03] se limitent au trois premiers des paragraphes suivants :

Performance de la structure de données Un octree est une structure de partitionnement spatial très efficace, qui nous permettra d'obtenir de très bonnes performances en termes de complexité des traitements. Ces traitements seront locaux aux cellules.

Simplification du nuage Discrétiser un nuage de cette manière le *simplifie*, au sens où la représentation discrète de la surface sous-jacente au nuage est plus grossière ; ceci limite donc l'influence du bruit au niveau des données acquises par un dispositif de mesure imparfait. Le niveau de détail est lié au niveau de subdivision des octree.

Hiérarchie Notre système prend en compte le caractère hiérarchique de l'arbre multi-résolution. Nous nous basons sur le résultat des traitements effectués au niveau s pour traiter les cellules du niveau $s + 1$.

Adjacence Le fait que ces octrees soient réguliers et que leur cellule racine soit identique implique la création d'un repère discret commun lié aux octrees. Ce repère nous permet d'introduire une relation d'adjacence entre les cellules. Le fait que notre algorithme soit très léger en terme de calcul (temps-réel) dépend beaucoup de cette adjacence.

Algorithmes définis localement Tous nos algorithmes sont définis par rapport à une cellule du premier octree (lié aux données $\mathcal{N}_1$) et le résultat de l'algorithme au niveau de subdivision précédent, lié au deuxième octree (lié aux données $\mathcal{N}_2$). Nous prenons également en compte la notion de voisinage des cellules. Ces algorithmes comprennent très peu de boucles et sont donc concis et simples à mettre en œuvre.

Régularité Les cellules des deux octrees sont de volume et de forme identiques. Le système ne fera pas face à des cas où deux cellules de taille très différentes viendraient à être comparées. Ce qui se traduirai en pratique, par exemple, par la comparaison d'une petite zone de la première surface (surface estimée par le nuage de point) avec une très grande zone de la deuxième, ce qui n'aurait que peu de sens.

Traitement d'image classique Notre but est de construire un flot de scène sur ce repère de voxels. Celui-ci étant régulier, tout comme une grille en trois dimensions, nous pouvons aisément appliquer nombre de traitements sur notre résultat tels que le calcul de gradient des données ou le filtrage. Nous exposerons ces post-traitements en partie 3.4.3.

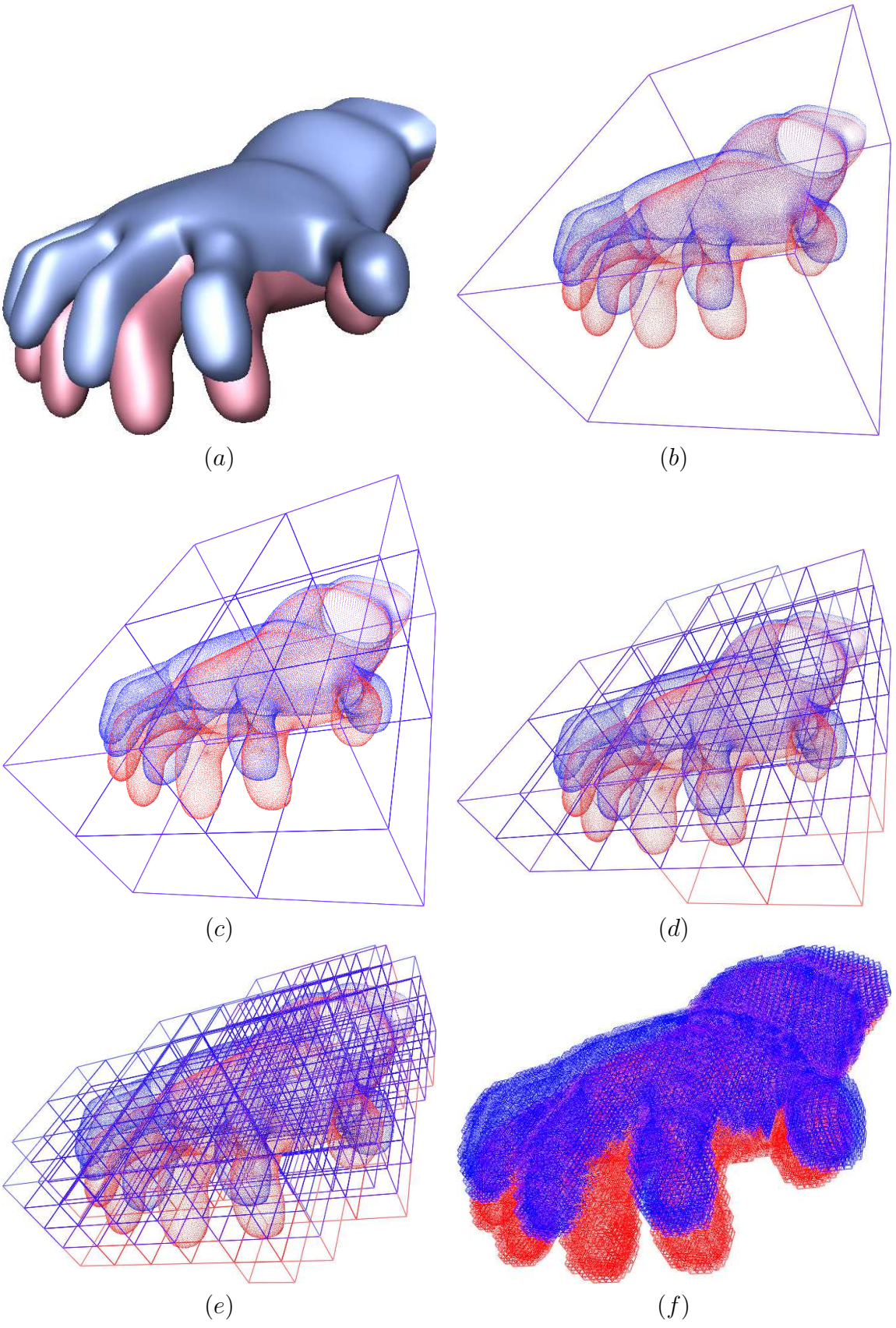


FIGURE 9 : *Exemple de voxélisation sur le modèle de synthèse d'une main, la surface rouge se déplace vers la surface bleue. De haut en bas et de gauche à droite : (a) visualisation des surfaces, (b) la cellule racine des deux arbres est confondue, (c) premier niveau de subdivision, (d) deuxième puis (e) troisième niveau, et nous visualisons enfin en (f) le niveau de subdivision* 6. *Les cellules des deux arbres se trouvent dans le même repère discret.*

Ainsi, soient deux octrees $\mathcal{O}_1$ et $\mathcal{O}_2$ dans lesquels nous plongeons respectivement les données de $\mathcal{N}_1$ et $\mathcal{N}_2$. Le niveau de subdivision maximal des octrees S est fixé, nous reviendrons sur cette question en partie 4. Pour la clarté de l'exposé, nous notons $\mathcal{U}_s$ les cellules non vides de $\mathcal{O}_1$ de niveau s, et $\mathcal{V}_s$ les cellules de non vides $\mathcal{O}_2$ à ce même niveau. L'appariement des cellules d'un niveau s est définit par :

$$m : \mathcal{U}_s \to \mathcal{V}_s \ .$$

Tout comme précédemment, puisque le nombre de cellules $\mathcal{U}_s$ peut être différent du nombre de cellules $\mathcal{V}_s$, nous considérons l'appariement des cellules

$$\mathcal{M} : \{(u, v) \mid u \in \mathcal{U}_s \, , v \in \mathcal{V}_s\} \ .$$

Les deux ensembles de voxels sont situés, à un niveau de subdivision s donné, dans un repère discret multi-résolution

$$\begin{cases} D : (i, j, k), & i, j, k \in \mathbb{N}^{3+} \\ i, j, k < 2^s \end{cases} \ .$$

Il est bien sûr possible de déterminer les coordonnées des éléments de D dans la base de coordonnées (O, x, y, z) de la scène, O étant une origine quelconque. De plus, certains voxels de $\mathcal{O}_1$ et $\mathcal{O}_2$ peuvent occuper la même position, c'est le cas de la cellule racine, mais il est a noter que les deux octrees ne contiennent pas les même sommets.

Nous définissons la notion d'adjacence entre les cellules par rapport à ce repère D. Celle-ci est similaire au voisinage de pixels dans une image : tout comme nous considérons la connexité 4 ou 8 en deux dimensions, nous considérerons la connexité 6, 18 ou 26 des voxels entre eux. Par contre, ce voisinage est légèrement plus complexe car nous analysons la superposition de deux octrees, donc deux couches de voxels. Nous introduisons alors deux types de voisinages :

- *Interne* : Ce voisinage utilise une relation d'adjacence entre les voxels d'un même octree.
- *Externe* : Ce voisinage utilise une relation d'adjacence entre les voxels de deux octrees différents.

Par exemple, le voisinage externe d'un voxel $u(i, j, k)$ est constitué du voisinage interne (donc le voisinage standard) du voxel $v(i, j, k)$. Afin de simplifier les algorithmes, nous considérons que le voisinage comprend également l'endroit du voxel lui-même, nous parlons donc de connexité 7, 19 ou 27 en trois dimensions ; cette notion est utile dans le cas du voisinage externe. Par la suite, nous utiliserons cette relation d'adjacence en prenant en compte la connexité 27. Du point de vue de l'implémentation, nous mettons en place efficacement cette adjacence en utilisant des indices fixés pour chaque voxels issus de la subdivision d'une cellule ; nous utilisons en l'occurrence le codage de Freeman.

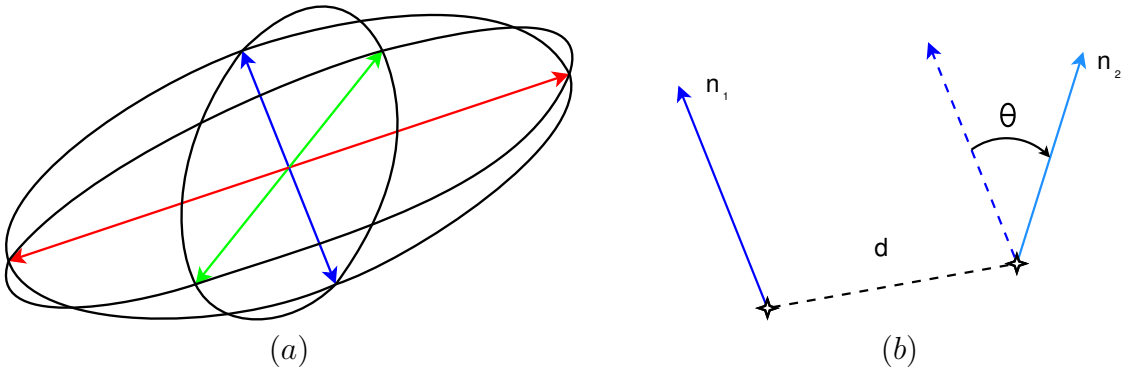


FIGURE 10 : *(a) Nous simplifions un nuage de points en considérant son centre de masse et ses trois axes principaux d'inertie, ici colorés dans un ordre de taille décroissante : rouge, vert et bleu. Le troisième axe est la normale de l'estimation planaire du nuage. En (b) nous illustrons le problème de la distance entre deux voxels : il s'agit d'une distance entre deux vecteurs orientés : les deux centres sont distants de d et les deux vecteurs forment l'angle θ.*

3.4 Construction du flot de scène

Comme exposé précédemment, le flot de scène est un champ de vecteurs qui conceptualise le mouvement de chaque élément d'un ensemble. Dans notre cas, tout comme pour les travaux existants, nous en distinguons deux : le flot des sommets du nuage $\mathcal{N}_1$ vers les sommets de $\mathcal{N}_2$, et le flot multi-résolution des voxels $\mathcal{U}_s$ vers les voxels $\mathcal{V}_s$. Nous nous intéressons pour l'instant à ce dernier, qui est entièrement défini par l'appariement $\mathcal{M}$. Cet appariement constitue la difficulté majeure rencontrée dans la conception de ce système, il est l'élément central du suivi de mouvement d'un nuage de points. Il s'agit de mettre en correspondance les voxels de deux octrees afin de les apparier. Or, ces voxels contiennent un ensemble de sommets épars, et la structure de l'octree n'a aucun rapport avec la géométrie sous-jacente aux nuages de points. Nous séparons ce problème en deux sous-parties :

- Comment comparer deux voxels situés à deux endroits quelconques du repère D ?
- Quels voxels devrions-nous comparer ? Lesquels devrions-nous apparier ?

Nous définirons en premier lieu une métrique, dans notre cas il s'agit d'une mesure de distance entre deux voxels. Puis nous détaillerons l'algorithme d'appariement récursif rapide et relativement simple que nous avons mis en place.

3.4.1 Une métrique appropriée

Nous cherchons l'appariement de voxels $\mathcal{M}$ qui minimise globalement la distance entre les deux nuages de points. La voxélisation des données implique la mise en correspondance locale des morceaux de surface décrits par les points contenus dans chaque voxel. Afin d'approximer un morceau de surface à partir d'un ensemble de points de l'espace, une solution parmi les plus simples est d'en estimer un plan tangent.

Une autre méthode simple serait d'approximer le nuage localement par une fonctionnelle quadratique (technique de *fitting*), voir [Boub 06]. À noter que les méthodes basées sur les maillages ont l'avantage de pouvoir aller plus loin que le plan tangent, car le maillage lui même est une meilleure approximation de la surface sous-jacente à un nuage de points. Ces considérations concernent la mise en correspondance géométrique des données : les études qui utilisent la donnée couleur peuvent quant à elles s'en servir efficacement pour une mise en correspondance photométrique, à la manière du traitement d'images classique, mais nous ne possédons pas cette information.

Ainsi, nous considérons chaque voxel comme une approximation locale du plan tangent de la surface étudiée. Tout comme pour le système [Agui 04] exposé précédemment, l'estimation d'un plan tangent à partir d'un ensemble de points passe par ses trois axes d'inertie, qui forment une ellipse voir Figure 10.a. En effet, si la surface sous-jacente est relativement planaire, alors les deux axes d'inertie principaux décrivent une approximation de ce plan, le troisième lui est normal.
Ces axes sont le résultat de l'analyse des moments géométriques de l'ensemble des points : le moment d'ordre 0 est le nombre de points, le moment d'ordre 1 est leur centre de masse, et les moments d'ordre 2 forment la matrice de covariance des données [Cheu 00]. Dans le cas d'un nuage $p_i \in \mathcal{N}_1, i \in [\![1, n_1]\!]$ nous avons :

$$\begin{aligned} M_0 &\sum_{i=1}^{n_1} 1 \ , \\ M_1 &= \frac{1}{M^0} \sum_{i=1}^{n_1} p_i \ , \\ M_2 &= \frac{1}{M^0} \sum_{i,j=1}^{n_1} p_i p_j \ , \end{aligned} \tag{1}$$

où $(u, v) \in \{x, y, z\}$. La matrice de covariance M est définie par ces moments d'ordre deux considérés selon ces trois coordonnées :

$$\begin{aligned} M &= 4 \begin{bmatrix} M_2|_{xx} & M_2|_{xy} & M_2|_{xz} \\ M_2|_{yx} & M_2|_{yy} & M_2|_{yz} \\ M_2|_{zx} & M_2|_{zy} & M_2|_{zz} \end{bmatrix} \\ &= V \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} V^T \ . \end{aligned} \tag{2}$$

Les vecteurs propres de M sont notés V et ses trois valeurs propres associées sont λ_0, λ_1 et λ_2. Si les valeurs propres sont ordonnées de façon décroissante, ceux-ci définissent les trois axes principaux d'inertie du nuage $\mathcal{N}_1$, les vecteurs $\{V_0, V_1, V_2\}$ sont normés et les valeurs propres définissent la taille de ces axes. En réalité nous utilisons la *Singular Value Decomposition* (*SVD*), dont le concept est très proche (implémentée efficacement au moyen de la *GSL* [5]).

5. *GNU Scientific Library* :
http ://www.gnu.org/software/gsl

Étant donnée notre structure régulière, il revient sensiblement au même de comparer deux plans tangents, relatifs à deux voxels, que de comparer les deux normales à ces plans, c'est dans cette optique que se sont basées nos travaux. Ces normales sont orientées par le troisième vecteur propre de V, et ont pour origine M_1 : le centre de masse des points. Leur norme λ_2 a de l'importance uniquement vis-à-vis de la planarité locale des données, nous préciserons cette notion en partie 4. Nous parlerons par la suite de "normale" comme étant la normale du plan tangent estimé des points contenus dans un voxel.
Notons par ailleurs que ces axes d'inertie sont orientés par les vecteurs propres V, qui forment un repère direct. Cependant les deux orientations du repère sont solutions de (2), elles sont dites chirales, il nous faut donc en choisir une. Dans notre cas, il nous faut régulariser l'orientation des normales afin de rendre leur orientation cohérente sur l'ensemble de la surface voxélisée. Nous effectuons ce traitement par propagation : nous choisissons une normale quelconque et nous propageons son orientation par voisinage de voxel à voxel (si le produit scalaire d'une normale voisine est négatif, alors nous l'inversons).

Nous cherchons une mesure de correspondance de deux voxels à travers une mesure de distance entre deux vecteurs normés dont les origines sont les centres de masse des données contenues dans ces derniers, voir Figure 10.b. Cette mesure est souhaitée faible si les données des voxels sont jugées proches, et grande si nous les jugeons éloignées.

Soit deux voxels $u \in \mathcal{O}_1$ et $v \in \mathcal{O}_2$. Nous considérons les vecteurs u : $(M_1(u), V_2(u))$ et v : $(M_1(v), V_2(v))$, que nous notons pour simplifier :

$$\left\{ \begin{array}{l} u \mapsto (g_u, n_u) \\ v \mapsto (g_v, n_v) \end{array} \right. .$$

Introduisons également la distance entre les deux centres de masse et l'angle mesuré entre les deux normales :

$$\left\{ \begin{array}{lcl} d & = & \| \overrightarrow{g_u g_v} \| \\ \cos\theta & = & \mid n_u \,.\, n_v \mid \end{array} \right. .$$

Notre mesure de distance prend en compte deux paramètres de dimension différentes, la distance et la différence angulaire. Ce à quoi nous ajoutons une information relative à la hiérarchie de notre structure de données, voir Figure 11. Nous posons l'heuristique suivante : si nous considérons pour un voxel u_s un vecteur d'appariement correct, alors les vecteurs d'appariements des voxels u_{s+1} issus de la subdivision de u_s serons proches. Nous pénalisons donc les écarts de mesures importantes entre ces deux données. Nous en déduisons donc notre mesure de correspondance δ entre deux voxels u_s et v_s :

$$\left\{ \begin{array}{ll} \delta(u_0, v_0) = & \| \overrightarrow{g_0 g_0} \| \\ \delta(u_s, v_s) = & \mid \| \overrightarrow{g_u g_v} \| - \delta(u_{s-1}, v_{s-1}) \mid \\ \varphi(u_s, v_s) = & 1 - \mid n_u \,.\, n_v \mid \end{array} \right. \qquad (3)$$

$$\Delta(u_s, v_s) = A\delta_{u_s v_s} + B\varphi_{u_s v_s} \ ,$$

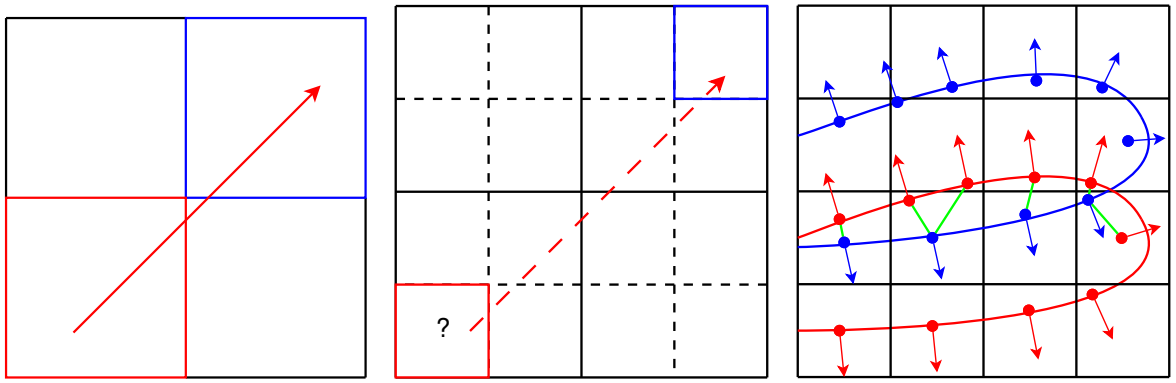


FIGURE 11 : *Illustration de l'apport d'information du niveau hiérarchique supérieur de l'octree. (a) Le voxel rouge est apparié avec le voxel bleu. Nous cherchons à apparier en (b) les voxels issus de la subdivision de ce dernier, nous pénalisons les appariements qui s'éloignerait du vecteur d'appariement déterminé précédemment. En (c), nous illustrons le fait qu'une simple mesure de distance provoquerai l'appariement de morceaux de surface d'orientation opposées, ce qui n'est pas souhaitable (les mauvais appariements sont colorés en vert).*

avec $\delta \in \mathbb{R}^+$ et $\varphi \in [0, 1]$. Les termes A et B sont des constantes de régularisation liées à l'hétérogénéité des deux dimensions de Δ : la distance et la variation angulaire. Dans nos tests, ces constantes sont fixées à 1 ; ce réglage donne la plupart du temps de bons résultats, mais il est difficile de lier automatiquement ces paramètres aux données que nous traitons. Nous aborderons en partie 4 deux alternatives à ce problème.

Par ailleurs, nous évoquions en partie 3.1 une raison pour laquelle une simple mesure de distance entre les centres de masse, en l'occurrence δ, était incomplète. Ceci est lié à l'orientation des surfaces sous-jacentes aux deux nuages de points, voir Figure 11.c. En effet, sans cette information d'orientation, les résultats de l'appariement ne pourraient être cohérents car nous pouvons rencontrer des cas de figure où deux morceaux de surfaces d'orientation opposée seraient très proches : ces régions ne doivent pas être appariées.

3.4.2 Algorithme d'appariement de voxels rapide

Les travaux existants précédemment cités et proches de notre méthode utilisent une recherche exhaustive des appariements entre les voxels, qui minimise une mesure de distance totale entre les deux octrees. Ce procédé est impraticable si le niveau de subdivision de l'octree est élevé. Si le niveau de subdivision est de l'ordre de 5 ou 6, n'importe quelle opération légèrement complexe sur les voxels entraînerai plusieurs minutes de temps de calcul. Nous introduisons ici un algorithme rapide pour l'appariement des voxels à des niveaux de subdivision élevés. Celui-ci est basé sur les notions de hiérarchie de la structure multi-résolution ainsi que sur le voisinage de ses cellules.

Nous introduisons ici la notion d'ensemble de vecteurs d'appariement :

$$\mathcal{X} : \{x = \overrightarrow{g_u g_v} \mid (u, v) \in \mathcal{M}\} \ .$$

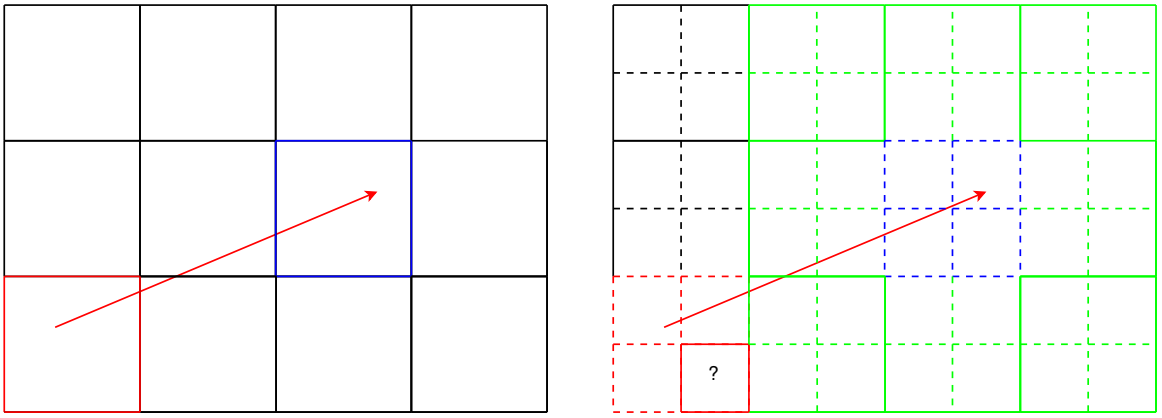


FIGURE 12 : *Illustration de l'influence de la hiérarchie de l'octree dans notre algorithme d'appariement. En (a), la cellule rouge est appariée avec la bleue au niveau de subdivision s. En (b), le niveau $s+1$ est tracé en pointillés, l'ensemble des cellules candidates à l'appariement de la cellule rouge est constitué des les cellules filles du voisinage de la cellule bleue du niveau s, nous les représentons en vert.*

Ces vecteurs ont pour origine le centre de masse g_u d'un voxel u et pour arrivée le centre de masse g_v du voxel v apparié à u. Les vecteurs $x \in \mathcal{X}$ forment un champ défini sur l'ensemble de l'octree $\mathcal{O}_1$ vers le deuxième octree $\mathcal{O}_2$: il s'agit du flot de scène des voxels, il nous servira à déterminer le flot de scène réel des données : $\mathcal{X}_p$.

Notre algorithme d'appariement est récursif, ses principales phases peuvent être résumées de la façon suivante :

1 - Initialisation Les deux cellules u_0 et v_0 sont appariées : $\mathcal{X} \leftarrow \overrightarrow{g_{u_0} g_{v_0}}$. Puis nous les subdivisions, et nous appliquons la phase 2 sur chacune d'elle.

2 - Appariement des cellules subdivisées Trouver la cellule v de $\mathcal{O}_2$ la plus proche parmi les candidates pour l'appariement de la cellule u de $\mathcal{O}_1$:

a. Parmi le voisinage externe de u, donc parmi 27 cellules de $\mathcal{O}_2$.
b. Parmi les cellules filles de l'appariement de niveau inférieur v_{s-1} de u_{s-1}, la cellule mère de u, ainsi que de son voisinage interne (donc également à l'intérieur de $\mathcal{O}_2$), voir une illustration en Figure 12.

3 - Subdivision des cellules u et v et récursion de l'algorithme Si le niveau maximal n'est pas atteint, donc $s < S$, nous subdivisont les cellules non-vides et nous relançons l'algorithme en phase 2 sur chacune d'entre elles.

La phase 2 utilise la hiérarchie de l'octree en considérant que le niveau $s-1$ est traité. La phase 3 nous assure que toutes les cellules non-vides d'un même niveau s sont subdivisées au même moment. Ceci est important vis-à-vis de nos relations d'adjacence : un algorithme récursif appliqué localement aux cellules (donc un parcours en profondeur de l'arbre) impliquerait des cas où certaines cellules voisines n'existeraient pas encore.

Voici une expression de la fonction d'appariement des cellules. Les octrees sont ici considérés comme déjà subdivisés jusqu'au niveau S. Nous présentons l'algorithme final du système en partie suivante, où nous exposerons la phase de régularisation du champ de vecteurs.

Fonction CellMap

Data : La cellule $u(i,j,k) \in \mathcal{O}_1$, son homologue dans $\mathcal{O}_2$: $v(i,j,k)$ et l'appariement du niveau précédent $\mathcal{M}_{s-1} : (u_{s-1}, v_{s-1})$.
Result : Un vecteur appariement $x_u = \overrightarrow{g_u g_w}$ où w est la cellule de $\mathcal{O}_2$ retenue pour l'appariement.

begin
  $\omega \leftarrow \emptyset$
  $d_{min} \leftarrow \infty$

  // Premières candidates : le voisinage externe de u
  $v^1 \leftarrow$**Neighbors**(v)
  for $cell \in v^1$ **do**
1   **if** $\Delta(u, cell) < d_{min}$ **and** $\mid n_u.n_v \mid \geq 0$ **then**
        $d_{min} \leftarrow \Delta(u, cell)$
        $\omega \leftarrow cell$

  // Deuxièmes candidates : les cellules filles des voisins de l'appariement v_{s-1}
  if $v_{s-1} \neq \emptyset$ **then**
    $v^1_{s-1} \leftarrow Neighbors(v_{s-1})$
    for $motherCell \in v^1_{s-1}$ **do**
      for $cell \in$ **Down**$(motherCell)$ **do**
2   **if** $\Delta(u, cell) < d_{min}$ **and** $\mid n_u.n_v \mid \geq 0$ **then**
            $d_{min} \leftarrow \Delta(u, cell)$
            $\omega \leftarrow cell$

  // Résultat s'il existe
  if $\omega \neq \emptyset$ **then**
    $x_u \leftarrow \overrightarrow{g_u g_w}$
  else
    $x_u \leftarrow \emptyset$

En **1** et **2** nous rejetons automatiquement tout appariement impliquant des orientations opposées au niveau des plans tangents.

Notons que si un voxel contient moins de trois points, nous ne pouvons pas estimer de plan tangent et dans ce cas, nous mesurons simplement la distance euclidienne des centres de masses.

Une autre considération vient des cas de figures où un voxel $v \in \mathcal{O}_2$ choisi pour minimiser la distance avec le voxel u a déjà été apparié à un voxel de $\mathcal{O}_1$. Cette question est liée à celles que nous aborderons en partie suivante, concernant les post-traitements appliqués au champ de vecteurs une fois que nous l'avons obtenu. Dans ces cas de figures, nous adoptons un point de vue proche de la problématique de la création de graphes biparties. Ces graphes considèrent deux ensembles de données ; leur but est de lier deux à deux chaque élément des deux ensembles. En règle générale, chaque élément doit être lié à un seul élément de l'autre ensemble tout en minimisant une fonctionnelle de coût (distance, poids etc.), on parle alors d'appariement (le terme précis est *matching*). Notre problématique est très proche de celle visant à créer ce type de graphe. Nous avons donc introduit dans notre système une adaptation de l'algorithme Hongrois [Kuhn 55] dont voici un exemple d'appariement entre $x \in X$ et $y \in Y$ minimisant une fonctionnelle $f(x, y)$:

Fonction StableMatching

Data : Les ensembles X et Y
Result : Les appariements $\mathcal{M} : \{(x, y) | x \in X, y \in Y\}$

```
begin
    M ← ∅
    while ∃ x | x ∉ M do
        y = min_y f(x, y)
        // Si y n'est pas apparié
        if y ∉ M then
            M ← (x, y)
        else
            // Donc il existe un appariement (x', y)
            if f(x, y) < f(x', y) then
                M ← (x, y)
                (x', y) est enlevé de M
            else
                x ne trouve pas d'appariement.
```

En d'autres termes, si une cellule candidate v pour l'appariement de u est déjà appariée à une autre cellule u', nous comparons la distance $\Delta(u, v)$ et $\Delta(u', v)$. Si $\Delta(u, v)$ est moindre alors nous remplaçons l'appariement précédent (u', v) par celui-ci. Au niveau de l'implémentation, nous avons mis cette procédure en place très simplement en stockant dans chaque cellule $v \in \mathcal{O}_2$ la distance Δ de son appariement éventuel.

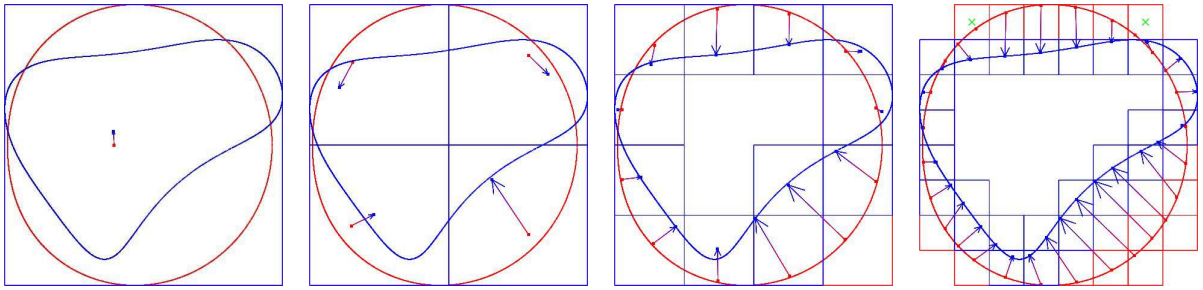


FIGURE 13 : *Exemple de flot de scène sur un exemple en deux dimensions, le nuage rouge se déplace vers le bleu. Notre algorithme hiérarchique est capable d'apparier des cellules éloignées sans augmenter le champ de recherche.*

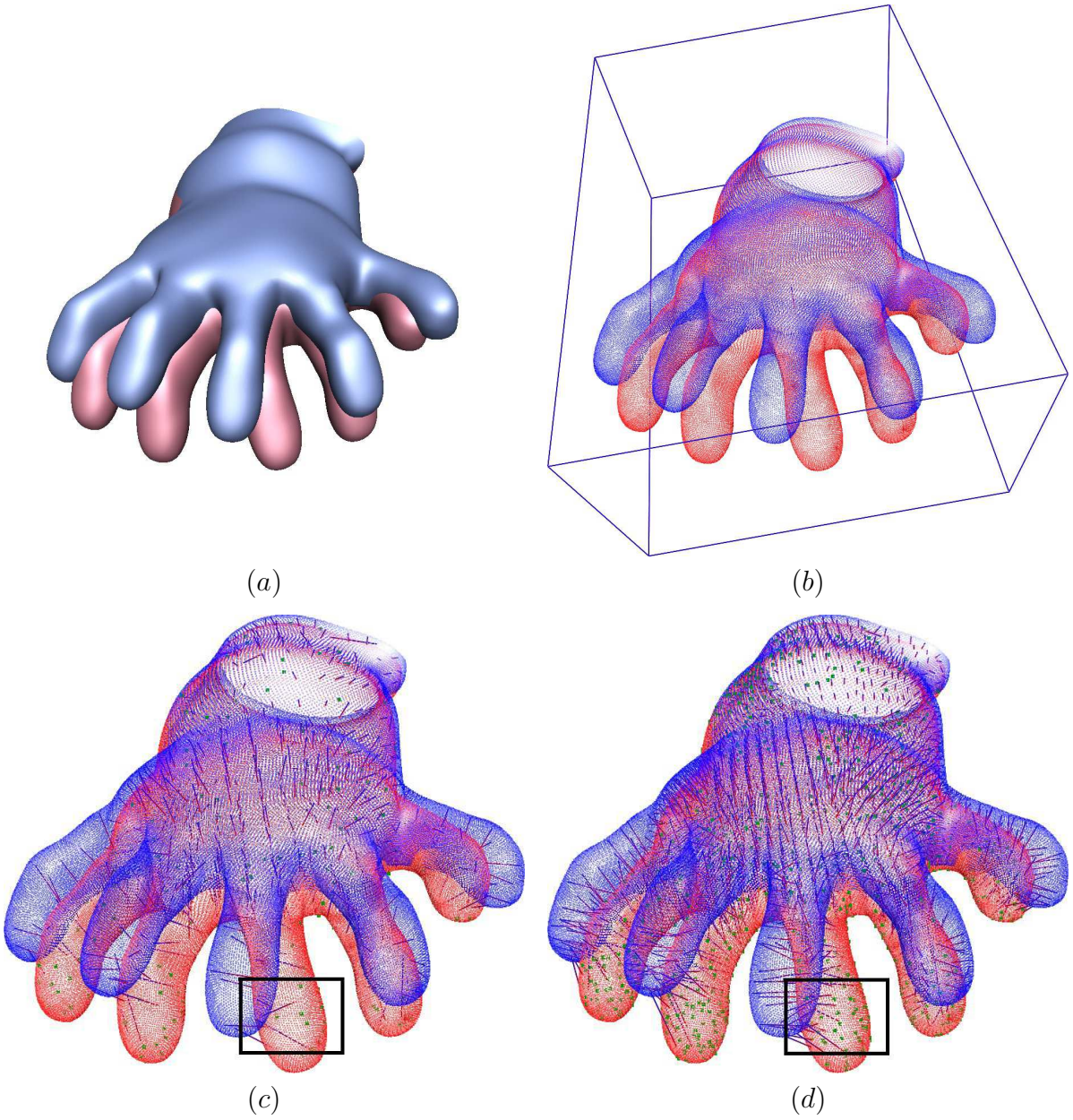


FIGURE 14 : *Exemple de flot de scène, visualisé au moyen de lignes à l'origine rouge et à l'arrivée bleue. (a) La surface rouge évolue vers la surface bleue. En (b) les deux nuages de points constituant nos données. En (c) le flot de scène pour un niveau de subdivision* 4 *: certaines cellules ne sont pas appariées (en vert). Ces cellules non appariées nuisent aux appariements des niveaux supérieurs, nous visualisons en (d) le niveau* 5.

Si l'appariement de la cellule de niveau inférieur u_{s-1} a bien été déterminé au moment de comparer $u_s(i,j,k)$ aux cellules candidates v_s, l'appariement pourra être envisagé à plusieurs voxels de distance de u, voir Figure 13. Il s'agit là justement de l'intérêt principal de la prise en compte multi-résolution de notre structure de données : le domaine de recherche des cellules candidates est constant, quelque soit le niveau de subdivision s. Notre algorithme est donc extrêmement rapide, mais il dépend fortement du résultat de l'appariement déterminé au niveau $s-1$. En effet si l'appariement de la cellule de niveau inférieur u_{s-1} n'a pas pu être fixé, alors la fonction `CellMap` limite sa recherche au voisinage externe de la cellule u_s. Ce domaine de recherche est très limité, et il est donc probable qu'à son tour, l'appariement de u_s ne puisse se faire, voir Figure 14. D'autre part, si l'appariement de u_{s-1} existe, l'algorithme considère qu'il est correct au sens des souhaits de l'utilisateur. Si celui-ci dénote un comportement peu souhaitable (ce qui est difficile à déterminer automatiquement), il est de même probable que les appariements des niveaux suivants aggravent le problème. C'est pourquoi nous ajoutons un ensemble de post-traitements au champ de vecteurs avant d'appliquer l'algorithme d'appariement sur le niveau de subdivision suivant.

3.4.3 Régularisation des appariements

Comme exposé précédemment en partie 3.3, le repère de voxels régulier D nous permet d'appliquer aisément certaines techniques classiques en traitement d'images. Nous prenons en compte deux problématique principales :

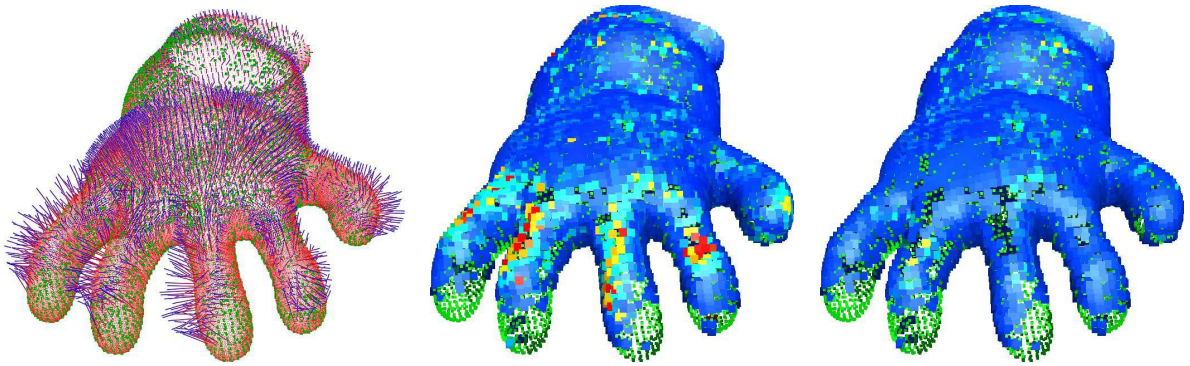
1 - Comment mesurer numériquement la qualité d'un appariement ?

2 - Comment trouver un appariement aux cellules pour lesquelles l'algorithme a échoué ?

Mesure numérique de la qualité d'un appariement

Il n'est pas trivial de mesurer de façon algorithmique la qualité globale d'un flot de scène ; l'utilisateur lui-même peut percevoir plusieurs appariements différents et néanmoins satisfaisants. Nous avons chercher à établir une mesure de qualité liée à l'homogénéité, ou la régularité, des vecteurs de déplacement.

Notre flot de scène est défini comme une grille tridimensionnelle de l'espace discret D où chaque voxel est un vecteur 3D. Une mesure de variation très classique est le gradient des données, il met notamment en évidence les contours présents dans une image dans le cas bidimensionnel. Nous proposons d'utiliser cette mesure comme l'évaluation numérique de la régularité de notre flot de scène. Dans notre cadre d'analyse, nous définissons ce gradient de la façon suivante pour tout voxel $u \in \mathcal{O}_1$:



(a) (b) (c)

FIGURE 15 : *Illustration de la mesure de gradient que nous introduisons sur l'exemple de la main (a). En (b) nous mesurons la norme du gradient de flot seuillée dans l'intervalle* $[0, 1]$ *: les couleurs vont du bleu (valeur* 0*) au rouge (valeur* 1*). Nous illustrons un exemple de seuillage de ces données en (c), pour un paramètre threshold* $= 50\%$*.*

$$G(u) = \quad x_u - \frac{1}{|u^1|} \sum_{w \in u^1} x_w \quad , \tag{4}$$

où u^1 sont les voxels du voisinage de u en connexité 27 pour lesquels nous avons déterminé un appariement, et x_w est le vecteur de déplacement associé au voxel w. Cette évaluation du gradient évalue donc la différence entre le vecteur x_u et la moyenne des vecteurs de son voisinage. En pratique, nous utiliserons la norme du vecteur $G(u)$ comme mesure numérique d'intensité du gradient du flot de scène mesuré en u, voir une illustration en Figure 15. Nous utilisons cette mesure d'homogénéité du flot afin d'identifier les cellules dont l'appariement est quantitativement éloigné de celui de ses voisines. Ce cas de figure dénote très probablement d'une erreur de l'algorithme puisque les objets que nous étudions se déplacent et se déforment continuement, l'ampleur des déformations est considérée grande par rapport à la taille de nos voxels.

Ainsi, nous proposons de filtrer ces cas d'appariements isolés. Nous introduisons un paramètre supplémentaire de seuil, noté *threshold*, au delà duquel l'appariement est jugé incorrect. Ce seuil est relatif à la mesure du gradient en un voxel u par rapport à la valeur maximale des gradients mesuré à un niveau de subdivision s de l'octree, il s'agit donc d'un pourcentage. La norme du gradient est proportionnelle à la taille des cellules de l'octree, ce rapport permet en pratique de marquer les cellules dont la mesure de gradient s'écarte de la moyenne pour un niveau de subdivision donné.

Cependant, tout comme les constantes d'homogénéité A et B (définies en partie 3.4.1), ce seuil est difficile à déterminer automatiquement pour chaque ensemble de nuages. Heureusement, notre système d'appariement étant temps-réel, l'utilisateur peut faire varier ce paramètre assez aisément. En l'état, nous recherchons toujours un moyen de rendre ce filtrage automatique.

Par ailleurs, nous avons étudié des méthodes simples de régularisation du flot telles que le filtrage passe bas relatif au gradient du champ de vecteurs. Ces méthodes rendent le flot de scène plus homogène, mais notre objectif reste l'appariement des voxels et non l'homogénéité simplement visuelle du champ. Notre système considère les vecteurs $\mathcal{V}$ du flot comme des liens entre les centres de masses des points contenus dans les voxels des deux octrees. Le fait de lisser (ou régulariser) ce champ n'apporte aucune autre information supplémentaire.

Nous proposons ci-dessous un algorithme de la fonction de seuillage, qui sera utilisé ensuite dans l'algorithme global de notre système.

Fonction CellFilter

Data : Les octrees $\mathcal{O}_1$ et $\mathcal{O}_2$, un voxel u, le niveau de subdivision choisi $s > 0$, l'ensemble d'appariements $\mathcal{M}$, $(u, v) \in \mathcal{M}$ et le champ de vecteurs d'appariements $\mathcal{X}_s$.
Result : Le champ $\mathcal{X}_s$ est modifié.

begin
  // La valeur de gradient maximale mesurée au niveau s de l'octree $\mathcal{O}_1$
  $maximum \leftarrow \max_{w \in \mathcal{O}_1^s} G(w)$
  // Seuillage
  if $G(u) > maximum \times threshold$ **then**
    (u, v) est enlevé de $\mathcal{M}$
    x_u est enlevé de $\mathcal{X}_s$

Assurance d'un appariement pour chaque cellule

Notre deuxième problème, une fois le champ de vecteurs construit pour un niveau s de l'octree $\mathcal{O}_1$, est liée aux cas où certaines cellules ne trouveraient aucune cellule candidate à l'appariement. Un tel comportement présent au niveau de subdivision s nuirait beaucoup aux résultats de l'algorithme pour les niveaux supérieurs à s. Nous proposons donc un moyen d'apparier de façon satisfaisante chaque voxel u du nuage.

La méthode que nous proposons est simple : pour chaque cellule u sans appariement, nous cherchons une cellule candidate parmi les appariements de son voisinage. Nous relâchons ainsi la contrainte des graphes biparties, qui pour le *matching* des éléments impose que chacun d'eux ne soit lié qu'à au plus un seul autre élément. Cette méthode à l'avantage d'être concise et de ne pas perturber le flot de scène. Cet appariement assure que le nouveau vecteur de déplacement ne sera pas éloigné du champ de vecteur local constitué par son voisinage.

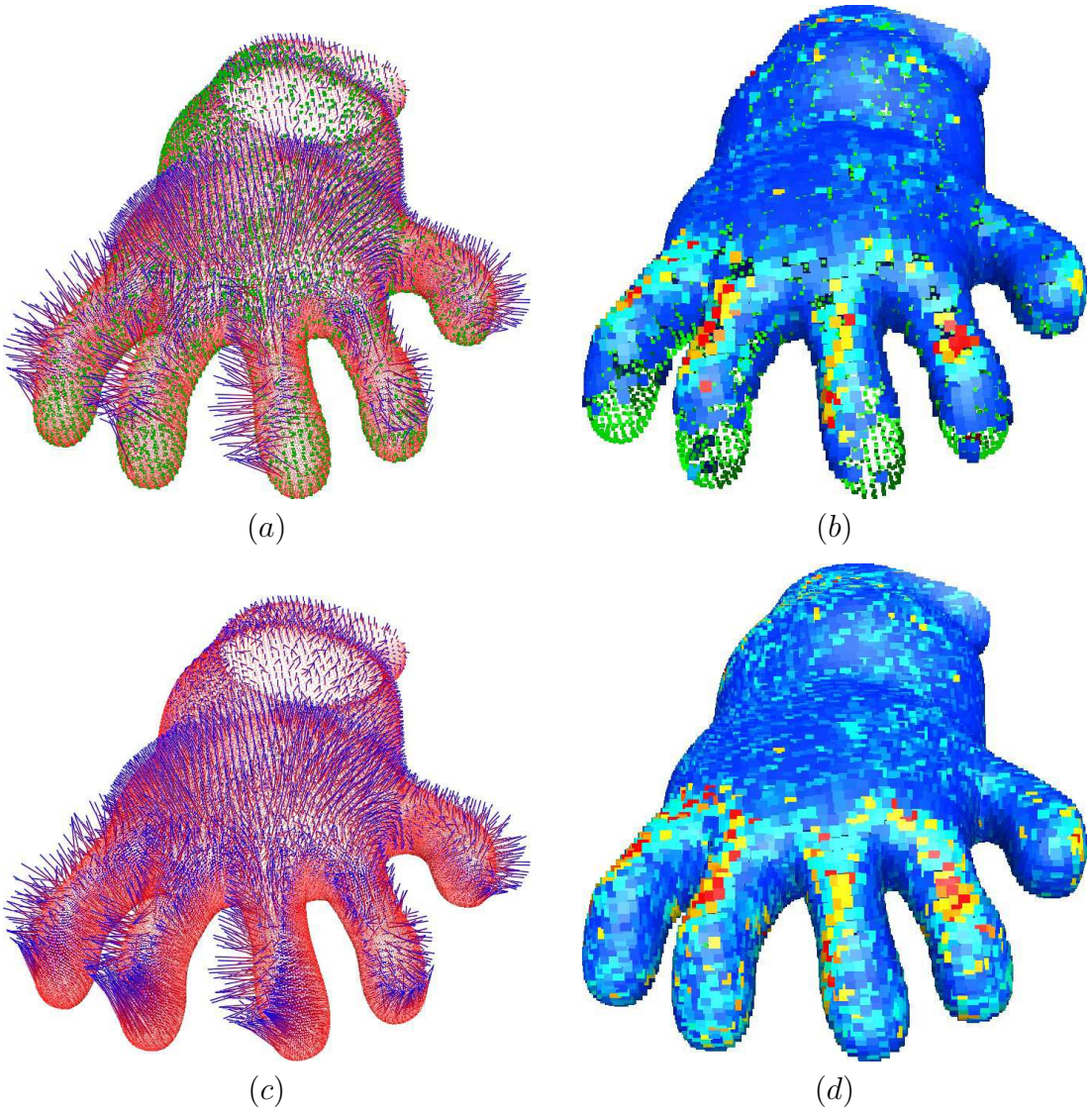


FIGURE 16 : *La diffusion nous permet d'apparier toutes les cellules de $\mathcal{O}_1$. En (a, b) l'exemple de main présenté précédemment. Nous appliquons en (c, d) notre algorithme de diffusion : toutes les cellules sont appariées.*

Fonction CellDiffusion

Data : Les octrees $\mathcal{O}_1$ et $\mathcal{O}_2$, un voxel u, le niveau de subdivision choisi $s > 0$ et le champ de vecteurs d'appariements $\mathcal{X}_s$.
Result : Le champ $\mathcal{X}_s$ est modifié.

begin
  // Chercher le meilleur appariement v pour u parmi ses voisins u^1
  $v \leftarrow \min_{w \in u^1} \Delta(u, w)$

  // Résultat s'il existe
  if $v \neq \emptyset$ **then**
    $\mathcal{X}_s \leftarrow (u, v)$

L'algorithme 1 décrit toutes les phases de notre système d'appariement de voxels. À noter que cette description algorithmique n'est pas optimisée pour une implémentation efficace.

Fonction RecursiveCellMap

Data : Les octrees $\mathcal{O}_1$ et $\mathcal{O}_2$, un voxel u, le niveau de subdivision choisi $s > 0$ et le champ de vecteurs d'appariements $\mathcal{X}_s$.
Result : Le champ de vecteurs d'appariements $\mathcal{X}_s$ du niveau s de l'octree $\mathcal{O}_1$.

begin
  if **Level**(u) $< s$ **then**
    RecursiveCellMap(**Down**(u))
  else
    $\mathcal{X}_s \leftarrow$ CellMap(u)

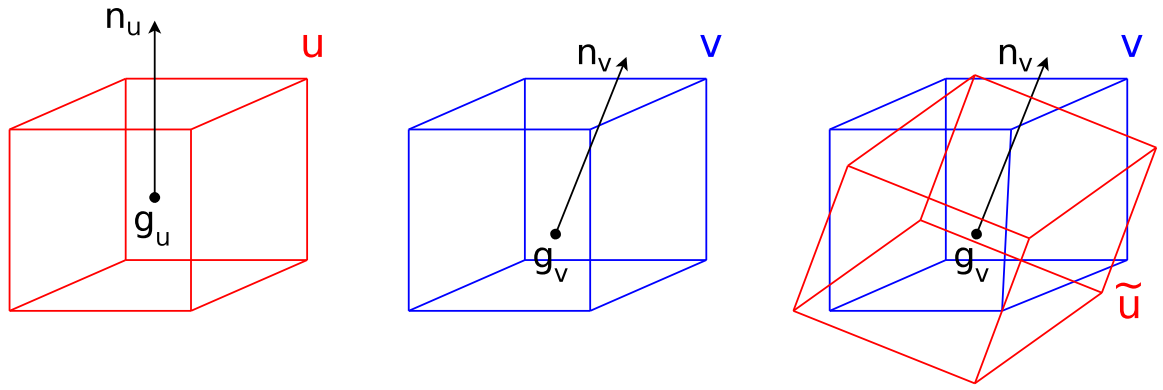
Algorithme 1: Appariement des voxels

Data : Les octrees $\mathcal{O}_1$ et $\mathcal{O}_2$, le niveau de subdivision S.
Result : Le champ de vecteurs d'appariements $\mathcal{X}$.

begin
  $\mathcal{X}_0 \longleftarrow (u_0, v_0)$
  for $s \leftarrow 0$ **to** $s \leq S$ **do**
    Subdivide($\mathcal{O}_1$)
    Subdivide($\mathcal{O}_2$)

  // Traitement niveau par niveau
  for $s \leftarrow 1$ **to** $s \leq S$ **do**
    $\mathcal{X}_s \leftarrow$ RecursiveCellMap(u_0, s)
    $\mathcal{X}_s \leftarrow$ RecursiveCellFilter($\mathcal{X}_s$, u_0, s)
    $\mathcal{X}_s \leftarrow$ RecursiveCellDiffusion($\mathcal{X}_s$, u_0, s)

Les deux fonctions *RecursiveCellFilter* et *RecursiveCellDiffusion* fonctionnent sur le même principe de récursion niveau par niveau que *RecursiveCellMap*, pour respectivement les fonctions *CellFilter* et *CellDiffusion*.



(a) (b) (c)

FIGURE 17 : *Recalage des données en vue de l'appariement des sommets : le voxel u en (a) est apparié au voxel v en (b). Nous recalons le voxel $u : (g_u, n_u)$ sur le repère géométrique local $\tilde{u} = \psi_{u,v}(v)$ de $v : (g_v, n_v)$.*

3.5 Utilisation du flot pour l'interpolation de deux nuages de points

À cet instant du traitement des données, nous possédons le champ de vecteurs d'appariement des voxels $\mathcal{X}$. Nous cherchons maintenant à établir les chemins liant chaque sommet du nuage $\mathcal{N}_1$ aux points du nuage $\mathcal{N}_2$; il s'agit du flot de scène que nous avons précédemment noté $\mathcal{M}_s$. La solution exposée dans [Cmol 03] propose d'effectuer un appariement des sommets localement à chaque paire de voxels appariés. Il s'agit donc, pour chaque couple de voxels appariés (u, v), de lier les sommets contenus dans u aux sommets contenus dans v. Nous considérons ici logiquement les voxels du niveau de subdivision le plus élevé des octrees.

Le fait est que nous possédons, en plus de cette information d'appariement entre les voxels u et v, une estimation de leur plan tangent : $u : (g_u, n_u)$ et $v : (g_v, n_v)$. La méthode d'appariement des sommets que nous introduisons ici est illustrée en Figure 17 :

1 - Recalage géométrique du voxel $u : (g_u, n_u)$ vers le voxel $v : (g_v, n_v)$. Le voxel u recalé est noté $\tilde{u} : (g_v, n_v)$.

2 - Appariement des sommets de $\tilde{u}$ avec les sommets de v par minimisation de la distance euclidienne globale. Nous obtenons alors l'appariement des sommets de u grâce aux sommets recalés de $\tilde{u}$.

Le deuxième point est proche de la minimisation de d_{min} exposée en partie 3.1. Cette méthode est envisageable car pour un niveau de subdivision S suffisamment élevé, chaque voxel ne contient qu'un faible nombre de sommets. Notons $\mathcal{M}_s(u, v)$ le flot de scène local aux points contenus dans le voxel u. Nous considérons donc que les points d'arrivée des vecteurs de ce flot sont entièrement contenus dans le voxel v, ce qui est discutable mais logique si l'appariement $\mathcal{X}$ est correctement formé. Notons également la fonction de recalage géométrique de voxel ψ, qui déplace les sommets de u vers le repère local de v, $\psi_{u,v} : u(g_u, n_u) \mapsto \tilde{u}(g_v, n_v)$. Par extension aux sommets de l'espace 3D contenus dans le voxel u nous avons $\tilde{p} = \psi_{u,v}(p)$ où $p \in u$ et $\tilde{p} \in \tilde{u}$.
Nous pouvons alors écrire :

$$\mathcal{M}_s(u, v) : \left\{ (p, q) \; \forall \; p \in u \mid q = \min_{r \in v} || \overrightarrow{\psi_{u,v}(p) r} || \right\} \; , \tag{5}$$

où $p, q, r \in \mathbb{R}^3$ sont des sommets. Ce recalage géométrique nous permet de mettre en correspondance deux petits nuages de points relativement proches l'un de l'autre. Autrement, une minimisation globale des distances entre les sommets produirait probablement un ensemble de vecteurs dont le point d'arrivée serait confondu en un point consistant un minimum de distance local.

La fonction de recalage ψ consiste en une translation selon le vecteur $d : \overrightarrow{g_u g_v}$ associée à une rotation tridimensionnelle $\varphi(n_u, n_v)$. Nous mettons efficacement en place la transformation angulaire des sommets de $\tilde{u}$ moyen d'un produit de quaternions défini par φ. Soient les points $p_u \in u_S$, ces points sont approximés par le couple de vecteurs (g_u, n_u). n_u étant le troisième axe d'inertie $V_2(u)$ des points p_u, ils sont approximés par le plan $(g_u, V_0(u), V_1(u))$. Afin d'exprimer les coordonnées de p_u dans la base (g_v, V_0^v, V_1^v), nous considérons le quaternion q défini par la rotation de $g_v + n_u$ vers $g_v + n_v$ autour de l'axe orthogonal $(g_v + n_u) \wedge (g_v + n_v)$. Ce quaternion défini à son tour une matrice de rotationR_q. Les points $\tilde{p}_u$ sont finalement exprimés de la façon suivante :

$$\tilde{p}_u = g_v + [R_q.(p_u - g_u)] \frac{||\overrightarrow{V_0^v}||}{||\overrightarrow{V_0^u}||} ,$$

et nous considérons la projection des points $\tilde{p}_u$ et p_v sur le plan $(g_v, V_0(v), V_1(v))$.

Nous appliquons enfin l'algorithme Hongrois précédemment exposé en section 3.4.2 afin d'obtenir l'appariement final des points des deux nuages. Cette fois, l'approche d'appariement naïve est est abordable puisque nous considérons que les cellules de niveau maximal contiennent peu de points.

4 Résultats et discussion

Cette section traite des choix parfois discutables que nous avons fixé concernant la conception de notre méthode. Ces sujets de discussions n'ont pour la plupart pas de rapport direct entre eux.

4.1 La question du morphing de nuages de points

Comme exposé précédemment durant l'état de l'art, le morphing de forme considère la transformation géométrique d'une forme vers une autre. Les techniques de morphing sont proches du suivi de forme (le tracking), car établir une relation de morphing entre deux objets revient plus ou moins directement à créer le mouvement d'un objet entre deux instants donnés.
La grande différence séparant ces deux domaines est la nature des objets analysés. Dans le cas du suivi, il s'agit d'un objet en mouvement mesuré dans l'espace à deux instants temporels fixés. Les techniques existantes en suivi s'intéressent notamment à caractériser le mouvements de certaines zones de l'objet ; dans le cas d'un humain il pourrait par exemple d'agir de ses membres. Le morphing, par contre, considère deux objets quelconques, et l'application de ces techniques est principalement axée sur le rendu visuel convaincant d'une déformation lisse entre eux.

Dans notre cadre d'analyse, nous pourrions probablement adapter notre système aux techniques de morphing en modifiant notre voxélisation. La régularité de celle-ci (les voxels ont tous la même taille et les deux octrees sont plongés dans le même repère) limite l'amplitude des mouvements de l'objet entre deux instants. Une voxélisation adaptative similaire à [Cmol 03] nous permettrait de lever cette contrainte, et ainsi de pouvoir caractériser un mouvement entre deux objets très différents. Nous perdrions par contre un certain nombre d'avantages propres à notre méthode initiale.

4.2 Le niveau de subdivision maximal S

Dans la phase de voxélisation de notre système (partie 3.3), nous considérons la subdivision régulière de voxels tridimensionnels. Nous fixons un niveau de subdivision S constant. Cette subdivision est uniforme, dans le sens où chaque voisin d'un voxel est un voxel de même niveau[6]. Nous n'avons cependant pas explicité le choix de cette constante S, ni n'avons exploré en profondeur ce choix d'uniformité de la subdivision. Cet aspect uniforme nous permet d'effectuer des traitements de voisinage de façon directe, cependant nous aurions pu modifier ces techniques dans l'optique d'une subdivision adaptative.
Nous avons considéré ces deux questions en prenant en compte l'aspect général des résultats produits par notre système. Il s'agit donc de choix qualitatifs, et une étude plus approfondie de ces considérations serait souhaitable. En l'état, nous fixons généralement $S = 6$ (ou $S = 7$). Ce choix subjectif produit des grilles tridimensionnelles de taille 64^3 (ou 128^3) voxels.

Un choix objectif pourrait logiquement être fait par rapport à la densité et la répartition des sommets du nuage dans chaque voxel. En effet, notre technique utilise l'appariement de plans tangents localement estimés (voir en partie 3.4.1). Hors cette estimation n'a pas de sens si le nombre de sommets considérés dans un voxel est faible, et à plus forte raison si leur nombre est inférieur à trois. Une condition d'arrêt de la subdivision adaptative serait alors liée à la qualité de cette estimation de planarité locale, donc à la mesure de la troisième valeur propre λ_2 de la matrice de covariance des sommets du voxel. En effet, cette mesure est faible par rapport aux autres valeurs propres si le nuage est correctement approximé par un plan, elle en est proche sinon (cette valeur λ_2 est nulle dans le cas où les sommets sont parfaitement coplanaires). Nous pourrions donc poser la condition booléenne de subdivision S_u d'un voxel u telle que :

$$\begin{cases} S_u = 1 & \text{si } \lambda_2/\lambda_1 < p \\ S_u = 0 & \text{sinon} \end{cases}, \tag{6}$$

où p est un pourcentage fixé relativement faible : $p = 10\%$ par exemple.

4.3 Alternatives à la métrique d'appariement de voxels Δ

Tel qu'exposé en partie 3.4.1, la mesure de similarité de deux voxels $\Delta(u, v)$ est fonction d'un terme de distance euclidienne et d'un terme angulaire. Ceux-ci sont liés aux centre de masses des voxels ainsi qu'à l'angle formé par leur deux normales associées. Ces termes n'ont pas la même dimension, nous les pondérons donc par deux constantes A

6. Celle-ci est régulière dans le sens où chaque voisin est un parallélépipède de même taille.

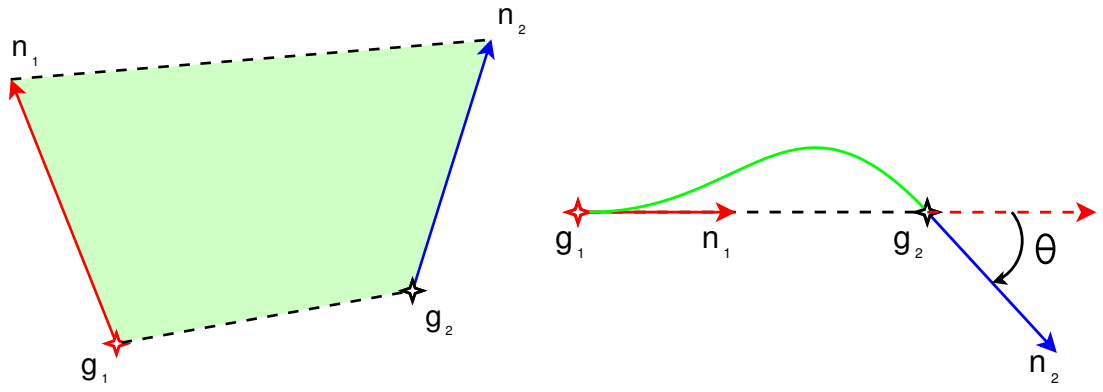


FIGURE 18 : *Deux alternatives à la mesure de distance Δ : (a) l'aire formée par le quadrilatère (g_1, n_1, n_2, g_2) et (b) la longueur de la courbe de Bézier créée par une composition de ces points.*

et B, que nous avons fixé à 1 de manière arbitraire à la seule vue des résultats corrects de nos tests. Plusieurs autres alternatives ont cependant été étudiées :

1. Une variable $B(s)$ liée au niveau de subdivision d'un voxel, au lieu d'une constante B.
2. La mesure de l'aire du quadrilatère formé par les deux vecteurs, créant ainsi une mesure homogène.
3. La longueur de la courbe de Bézier formée par ce quadrilatère de contrôle.

La première technique est basée sur l'idée de planarité locale des points du nuage dans chaque voxel que nous comparons. Aux premières itérations de subdivision des octrees, les nuages locaux sont très probablement mal approximés par des plans. Ceci car nos objets ne sont pas contraints aux formes planaires. Ainsi, cette fonction $B(s)$ à pour objet de limiter fortement l'influence du terme angulaire de Δ lorsque la planarité locale du nuage est faible (puisque ce terme angulaire définit la différence d'orientation des plans estimés). Cependant, cette méthode s'est avérée toute aussi subjective que l'utilisation des deux constantes, nous avons alors abandonné cette idée.

La deuxième idée est exposée en Figure 18.a : il s'agit d'unifier les dimensions de distance et d'angle en considérant l'aire formée par les deux centres de masses g et les deux normales n.
Cependant, cette mesure donne de très mauvais résultats.

Enfin, la troisième alternative est de considérer une courbe de Bézier influencée par la distance g_1g_2 autant que par la mesure angulaire θ, voir Figure 18.b. Nous utilisons pour cela une combinaison géométrique simple des quatre sommets d'intérêt. Cette mesure de longueur est implémentée efficacement au moyen de la technique [Guen 90].
Malheureusement, cette technique fonctionne également assez mal, pour une raison tout aussi indéterminée.

Références

[Agui 04] E. D. Aguiar, C. Theobalt, M. Magnor, H. Theisel, and H. peter Seidel. "M^3 : Marker-free Model Reconstruction and Motion Tracking from 3D Voxel Data". In : *In Proceedings of Pacific Graphics 2004*, pp. 101–110, 2004.

[Agui 07] E. de Aguiar, C. Theobalt, C. Stoll, and H. P. Seidel. "Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture". *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.

[Alex 02] M. Alexa. "Recent Advances in Mesh Morphing". *Computer Graphics Forum*, Vol. 21, pp. 173–196, 2002.

[Alex 03] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. "Computing and Rendering Point set surfaces". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 1, pp. 3–15, January 2003.

[Alex 04] M. Alexa, M. Gross, M. Pauly, H. Pfister, M. Stamminger, and M. Zwicker. "Point-Based Computer Graphics". In : *SIGGRAPH 2004 Course Notes*, 2004.

[Arci 10a] R. Arcila, K. Buddha, S., F. Hétroy, F. Denis, and F. Dupont. "A Framework for motion-based mesh sequence segmentation". In : *International Conference on Computer Graphics, Visualization and Computer Vision, WSCG*, Plzen, Czech Republic, Feb. 2010.

[Arci 10b] R. Arcila, K. Buddha, S., F. Hétroy, F. Denis, and F. Dupont. "A Framework for motion-based mesh sequence segmentation". In : *International Conference on Computer Graphics, Visualization and Computer Vision, WSCG*, Plzen, Tchèque, République, Feb. 2010.

[Bao 05] Y. Bao, X. Guo, and H. Qin. "Physically based morphing of point-sampled surfaces". *Computers Animation Virtual Worlds*, Vol. 16, 2005.

[Beau 95] S. S. Beauchemin and J. L. Barron. "The computation of optical flow". *ACM Comput. Surv.*, Vol. 27, pp. 433–466, September 1995.

[Boub 06] T. Boubekeur, W. Heidrich, X. Granier, and C. Schlick. "Volume-Surface Trees". *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2006)*, Vol. 25, No. 3, pp. 399–406, 2006.

[Bron 07a] E. M. Bronstein, S. Member, M. M. Bronstein, S. Member, R. Kimmel, and S. Member. "Calculus of non-rigid surfaces for geometry and texture manipulation". *IEEE Transactions on Visualization and Computer Graphics*, pp. 902–913, 2007.

[Bron 07b] E. M. Bronstein, S. Member, M. M. Bronstein, S. Member, R. Kimmel, and S. Member. "Calculus of non-rigid surfaces for geometry and texture manipulation". *IEEE Transactions on Visualization and Computer Graphics*, pp. 902–913, 2007.

[Cagn 10] C. Cagniart, E. Boyer, and S. Ilic. "Probabilistic Deformable Surface Tracking From Multiple Videos". In : *11th European Conference on Computer Vision*, Heraklion, Grèce, 2010.

[Carr 03] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. "Free-viewpoint video of human actors". In : *ACM SIGGRAPH 2003 Papers*, pp. 569–577, ACM, New York, NY, USA, 2003.

[Cheu 00] G. K. M. Cheung, T. Kanade, J. Y. Bouguet, and M. Holler. "A Real Time System for Robust 3D Voxel Reconstruction of Human Motions". *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, Vol. 2, p. 2714+, 2000.

[Cmol 03] L. . Cmolìk and M. Uller. "Point Cloud Morphing". In : *In Proceedings of the 7th Central European Seminar on Computer Graphics*, pp. 97–105, 2003.

[Cora 06] S. Corazza, L. M$\frac{1}{4}$ndermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. "A markerless motion capture system to study musculoskeletal biomechanics : visual hull and simulated annealing approach". *Annals of Biomedical Engineering*, Vol. 34, No. 6, pp. 1019–29, 2006.

[Deca 00] D. Decarlo and D. Metaxas. "Optical flow constraints on deformable models with applications to face tracking". *International Journal of Computer Vision*, Vol. 38, pp. 99–127, 2000.

[Demp 77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the royal statistical society, Series B*, Vol. 39, No. 1, pp. 1–38, 1977.

[Fran 10] J.-S. Franco, L. Guan, E. Boyer, and M. Pollefeys. "Flot d'occupation 3D à partir de silhouettes latentes". In : *Reconnaissance de Forme et Intelligence Artificielle*, Caen, France, Jan. 2010.

[Gros 07] M. Gross and H. Pfister, Eds. *Point-Based Graphics*. Elsevier, 2007.

[Guen 90] B. Guenter and R. Parent. "Computing the arc length of parametric curves". *IEEE Computer Graphics and Applications*, Vol. 10, pp. 72–78, 1990.

[Guo 05] X. Guo and H. Qin. "Real-time meshless deformation". *Journal of Visualization and Computer Animation*, Vol. 16, pp. 189–200, 2005.

[Hopp 92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Surface reconstruction from unorganized points". In : *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pp. 71–78, ACM, New York, NY, USA, 1992.

[Keri 06] R. Keriven and O. Faugeras. "Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score". *The International Journal of Computer Vision*, Vol. 72, p. 2007, 2006.

[Kuhn 55] H. W. Kuhn. "The Hungarian Method for the assignment problem". *Naval Research Logistics Quarterly*, Vol. 2, pp. 83–97, 1955.

[Laza 98] F. Lazarus and A. Verroust. "Three-dimensional metamorphosis : a survey". *The Visual Computer*, Vol. 14, No. 8/9, pp. 373–389, 1998.

[Lins 01] L. Linsen. "Point Cloud Representation". 2001.

[Lore 87] 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*, pp. 163–169, ACM, New York, NY, USA, 1987.

[Myro 10] A. Myronenko and X. Song. "Point Set Registration : Coherent Point Drift". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 12, pp. 2262–2275, 2010.

[Neum 02] J. Neumann and Y. Aloimonos. "Spatio-Temporal Stereo Using Multi-Resolution Subdivision Surfaces". *Int. J. Comput. Vision*, Vol. 47, pp. 181–193, April 2002.

[Preu 03] T. Preusser and M. Rumpf. "Extracting Motion Velocities from 3D Image Sequences and Coupled Spatio-Temporal Smoothing". In : *In Proc. SPIE Visual Data Analysis*, pp. 3–88722, 2003.

[Rusi 00] S. Rusinkiewicz and M. Levoy. "QSplat : A Multiresolution Point Rendering System for Large Meshes". In : *Proceedings of ACM SIGGRAPH 2000*, pp. 343–352, July 2000.

[Salz 07] M. Salzmann, J. Pilet, S. Ilic, and P. Fua. "Surface deformation models for non-rigid 3âd shape recovery. to appear". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, pp. 1481–1487, 2007.

[Sieb 07] M. von Siebenthal, G. Szekely, U. Gamper, P. Boesiger, A. Lomax, and P. Cattin. "4D MR imaging of respiratory organ motion and its variability". *Physics in Medicine and Biology*, Vol. 52, No. 6, pp. 1547–1564, March 2007.

[Star 05a] J. Starck, G. Miller, and A. Hilton. "Video-Based Character Animation". *ACM SIGGRAPH Symposium on Computer Animation (SCA)*, pp. 49–58, 2005.

[Star 05b] J. Starck and A. Hilton. "Spherical Matching for Temporal Correspondence of Non-Rigid Surfaces". In : *International Conference on Computer Vision*, pp. 1387–1394, 2005.

[Star 07] *Correspondence labelling for wide-timeframe free-form surface matching*, 2007.

[Sund 03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. "Skeleton Based Shape Matching and Retrieval". In : *Proceedings of the Shape Modeling International 2003*, pp. 130–, IEEE Computer Society, Washington, DC, USA, 2003.

[Tang 08] J. W. H. Tangelder and R. C. Veltkamp, Eds. *Multimedia Tools and Applications*. Vol. 39, Springer, 2008.

[Tree 01] G. M. Treece, R. W. Prager, and A. H. Gee. "Volume-based three-dimensional metamorphosis using region correspondence". *Visual Computer*, Vol. 17, pp. 397–414, 2001.

[Vara 08] K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. "Temporal Surface Tracking using Mesh Evolution". In : D. A. Forsyth, P. Torr, and A. Zisserman, Eds., *10th European Conference on Computer Vision, ECCV'08, October, 2008*, pp. 30–43, Springer, Marseille, France, Oct. 2008.

[Vedu 00] S. Vedula, S. Baker, S. Seitz, and T. Kanade. "Shape and Motion Carving in 6D". In : *In IEEE Conference on Computer Vision and Pattern Recognition*, pp. 592–598, 2000.

[Vedu 99] S. Vedula, S. Baker, P. Rander, R. T. Collins, and T. Kanade. "Three-Dimensional Scene Flow". In : *International Conference on Computer Vision*, pp. 722–729, 1999.

[Xiao 04] C. Xiao, W. Zheng, Q. Peng, and A. R. Forrest. "Robust morphing of point-sampled geometry : Research Articles". *Comput. Animat. Virtual Worlds*, Vol. 15, pp. 201–210, July 2004.

[Xie 08] J. Xie, P. ann Heng, and M. Shah. "Shape matching and modeling using skeletal context". *Pattern Recognition*, Vol. 41, pp. 1756–1767, 2008.

[Yoon 10] S. M. Yoon. *Markerless Motion Analysis in Diffusion Tensor Fields and Its Applications*. PhD thesis, TU Darmstadt, June 2010.

[Zaha 11] A. Zaharescu, E. Boyer, and R. P. Horaud. "Topology-Adaptive Mesh Deformation for Surface Evolution, Morphing, and Multi-View Reconstruction". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 4, pp. 823 – 837, April 2011.

[Zhan 99] D. Zhang and M. Hebert. "Harmonic Maps and Their Applications in Surface Matching". In : *Computer Vision and Pattern Recognition*, pp. 2524–2530, 1999.

[Zige 02] G. Zigelman, R. Kimmel, and N. Kiryati. "Texture Mapping Using Surface Flattening via Multidimensional Scaling". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, pp. 198–207, 2002.