Université de Montréal

Reconstruction volumétrique par l'algorithme du flot maximum dans un graphe

par

Catherine Proulx

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

20 janvier 2005

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé :

Reconstruction volumétrique par l'algorithme du flot maximum dans un graphe

présenté par

Catherine Proulx

a été évalué par un jury composé des personnes suivantes:

Jean Meunier
(président-rapporteur)

Sébastien Roy
(directeur de recherche)

Pierre Poulin
(membre du jury)

Mémoire accepté le 11 mars 2005

RÉSUMÉ

Nous proposons une méthode passive de reconstruction 3D basée sur l'algorithme du flot maximum dans un graphe. À partir d'images calibrées d'un objet, on cherche globalement le modèle 3D le plus probable étant donné les contraintes de photo-cohérence – la couleur projetée d'un point dans toutes les images est cohérente avec le modèle d'illumination – et de continuité spatiale des surfaces. Cette recherche se fait radialement à partir du centre du volume de reconstruction, ce qui impose une contrainte topologique aux objets pouvant être reconstruits. L'occlusion est gérée itérativement: on suppose initialement qu'il n'y a pas d'occlusion et on raffine successivement l'information de visibilité à partir des premières reconstructions. La méthode développée permet une reconstruction rapide et robuste d'objets simples même dans des conditions difficiles.

Mots-clés: vision par ordinateur, stéréoscopie, reconstruction 3D, méthodes de graphes, flot maximum, photo-cohérence, occlusion

ABSTRACT

We present a passive 3D reconstruction technique based on the maximum-flow algorithm. Starting with a set of calibrated images, we globally search for the most probable 3D model given the photoconsistency constraint – i.e., the projected color of any point in the scene is coherent with the lighting model – and the spatial continuity constraint. This search is done radially from the origin of the reconstruction volume; therefore we impose a topology constraint on the objects that can be reconstructed. Occlusion is managed iteratively: we initially assume that there is no occlusion and we refine the visibility information based on the first solutions. Our method is fast and robust when dealing with simple objects, even in difficult conditions.

Keywords: computer vision, stereoscopy, 3D reconstruction, graph-based, max-flow, photo-consistency, occlusion

TABLE DES MATIÈRES

LISTE DES FIGURES

REMERCIEMENTS

Je voudrais tout d'abord remercier le CRSNG et l'Université de Montréal pour leur soutien financier.

Merci à tous ceux qui m'ont guidée, aidée et supportée (dans tous les sens du terme) au cours des deux dernières années: le personnel et les professeurs du département, mes collègues de laboratoire ainsi que ma famille et mes proches, spécialement Aris qui a été à mes côtés tout au long de ce projet.

Finalement, un merci tout particulier à Sébastien pour son enthousiasme, son soutien et sa confiance inébranlable.

Chapitre 1

INTRODUCTION

1.1 Introduction

La reconstruction tridimensionnelle est un des problèmes centraux de la vision par ordinateur. Le problème consiste à extraire un modèle 3D virtuel d'une scène réelle à partir d'images de celle-ci. Ce modèle pourra ensuite être stocké tel quel ou encore être manipulé dans un but précis. Les applications d'une telle technologie sont innombrables, allant de l'archivage d'artéfacts archéologiques à la construction d'éléments de décor pour les jeux vidéos, en passant par la "photographie 3D" grand public et l'imagerie médicale.

Il existe déjà des systèmes permettant de numériser un objet pour en recouvrir un modèle 3D. Ces systèmes, dits de reconstruction active, fonctionnent selon le principe de la lumière structurée: des motifs de lumière sont projetés sur l'objet à reconstruire, puis capturés par plusieurs caméras placées autour de lui. Le système peut facilement mettre en correspondance les points ainsi éclairés dans les différentes images, et retrouver leur position dans l'espace par triangulation.

Les systèmes de reconstruction par lumière structurée sont efficaces, mais ils comportent plusieurs inconvénients majeurs qui ont empêché jusqu'à maintenant leur usage répandu. Le premier de ces inconvénients est leur prix: les systèmes de lumière structurée demandant de l'équipement très spécialisé, leur coût les rend hors de portée de la plupart des utilisateurs potentiels. De plus, certains sujets se prêtent mal à la reconstruction active parce qu'on ne peut contrôler leur illumination (ex: paysage, gratte-ciel...) ou qu'ils ne peuvent pas rester immobile pendant tout le processus

(danseur en mouvement, animal...). Il serait donc avantageux de développer un système permettant de capturer instantanément plusieurs images d'une scène avec des appareils-photo bon marché, puis d'analyser ces images en différé pour en dégager une structure 3D.

C'est ce que propose la reconstruction passive. En éliminant l'interaction avec la scène, le problème devient cependant beaucoup plus difficile à résoudre, puisque la mise en correspondance des points de la scène ne se fait plus de façon simple grâce aux motifs lumineux. On se retrouve avec un problème sous-déterminé, où plusieurs scènes peuvent expliquer les images analysées. On doit donc trouver la scène la plus probable parmi celles-ci.

L'objectif de ce projet est de développer une nouvelle méthode de reconstruction passive, basée sur les graphes, qui s'avère plus robuste que les techniques existantes. Notre stratégie consiste à étendre une méthode établie de stéréoscopie classique - la stéréoscopie par flot maximum dans les graphes de Roy et Cox [28] - au domaine de la reconstruction 3D. Cette méthode introduit une contrainte de lissage permettant de discriminer entre les différentes solutions possibles et de retrouver un modèle raisonnablement peu bruité de la scène, et ce même dans des conditions non idéales.

Nous commencerons par définir plus précisément le problème et par survoler les techniques de reconstruction et de stéréoscopie existantes pour situer notre recherche. Nous présenterons ensuite, au chapitre 3, le projet de recherche à proprement dit qui peut être divisé en quatre grandes sections: le calcul de la fonction de coût, la recherche de solution, la gestion des occlusions, et les raffinements de l'algorithme. Finalement, les chapitres 4 et 5 présenteront les résultats obtenus et les conclusions qu'on peut en tirer.

1.2 Définition du problème

Le problème général de la reconstruction 3D peut être défini comme suit: soit un ensemble d'images d'une scène réelle, trouver un modèle synthétique en trois dimensions représentant le plus fidèlement possible cette scène.

La première contrainte importante, c'est que les images doivent être calibrées. Ceci signifie qu'on peut prédire à quel endroit des images sera projeté n'importe quel point du monde. Ce calibrage prend la forme d'une matrice de projection permettant de passer du domaine 3D au domaine 2D. La méthode utilisée pour la calibration importe peu, en autant que celle-ci soit précise à quelques pixels près. On doit aussi connaître un volume englobant toute la scène: le volume de reconstruction.

La solution pourra être représentée sous plusieurs formes. La plus typique de ces formes est probablement la représentation par voxels puisque c'est celle qui est à la base de la plupart des algorithmes de reconstruction classiques. Notre approche étant plus appropriée à une représentation *b-rep* (*boundary representation*), la sortie de notre algorithme sera plutôt un maillage de triangles. Cette représentation a l'avantage d'être directement utilisable dans la plupart des applications 3D, alors qu'une représentation par voxels devra généralement être interprétée par un algorithme comme celui des *marching cubes* [22].

Dans le cadre de ce projet, nous avons choisi de limiter légèrement la portée du problème à résoudre en restreignant la classe des objets pouvant être reconstruits aux objets de topologie radiale. Nous définissons un objet de topologie radiale comme suit: étant donné un point c placé au centre de l'objet, un objet a une topologie radiale si toutes les droites allant du point c vers l'extérieur de l'objet ne croisent la surface qu'une seule fois (figure 1.1). On doit donc choisir le volume de reconstruction de manière à ce que son origine soit située à l'intérieur de l'objet, en un point c permettant à cet objet de respecter la contrainte de topologie radiale. Nous verrons à la section 3.6.2 qu'il serait cependant possible d'assouplir cette contrainte en proposant

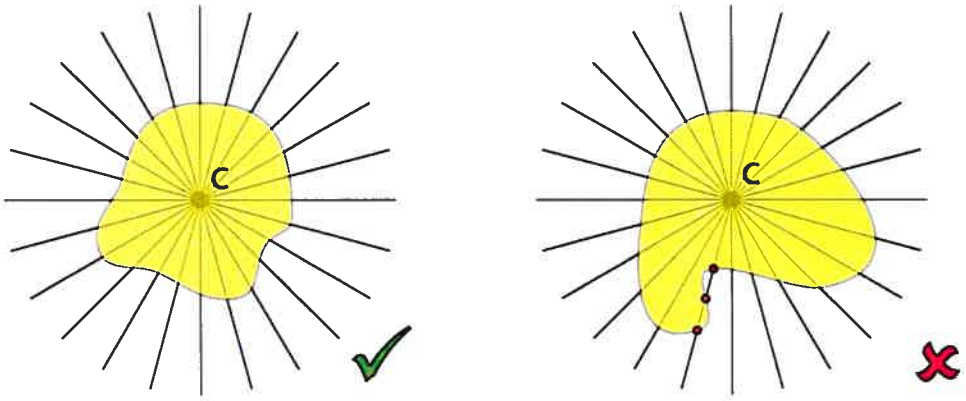


Figure 1.1: Un objet a une topologie radiale si les rayons partant du point c ne croisent la surface qu'une seule fois.

une solution initiale au problème.

Nous posons aussi plusieurs hypothèses simplificatrices pour faciliter la reconstruction de la scène. La première de ces hypothèses est que la surface à reconstruire est lambertienne. Une *surface lambertienne* est une surface parfaitement matte dont l'intensité lumineuse est perçue de la même manière par différents observateurs. En d'autres mots, un point d'une surface lambertienne a toujours la même couleur peu importe d'où on le regarde. Cette hypothèse est fréquemment utilisée en vision parce qu'elle évite de connaître la position de l'observateur ainsi que le modèle d'illumination de la scène et de recourir à des modèles physiques complexes. En réalité, aucune surface n'est parfaitement lambertienne, mais il a été établi empiriquement que cette hypothèse donne de bons résultats même si la scène à reconstruire comporte des matériaux légèrement spéculaires. Les reflets présents sur cette surface pourront être interprétés et traités comme du bruit, et ne devraient pas affecter trop fortement les résultats.

L'autre hypothèse est que la surface est suffisamment texturée pour pouvoir mettre en correspondance les points vus par les différentes caméras. Si les caractéristiques de la surface et l'illumination font que l'objet a une couleur parfaitement uniforme, il

sera impossible de reconstruire l'objet avec notre méthode. Finalement, on suppose aussi que les caméras sont réparties plus ou moins uniformément autour de la scène. La configuration exacte des caméras est arbitraire, en autant que la plupart des points de la surface soient vus par au moins deux caméras.

Chapitre 2

THÉORIE DE LA RECONSTRUCTION 3D

L'approche que nous proposons consiste à prendre une méthode de stéréoscopie classique – la méthode de stéréoscopie par flot maximum dans les graphes de Roy et Cox [28] – et à l'étendre au domaine de la reconstruction volumétrique. Nous commencerons par définir plus précisément les bases théoriques nécessaires à la compréhension de ce mémoire, puis ensuite faire le survol des articles majeurs en stéréoscopie et en reconstruction. Nous présenterons finalement quelques méthodes hybrides qui tentent, comme la nôtre, de tirer le meilleur de ces deux techniques. Ce survol devrait permettre de démontrer que les approches globales de reconstruction volumétrique présentent un intérêt certain, et que notre méthode s'insère effectivement dans un domaine peu étudié mais néanmoins prometteur de la reconstruction 3D.

2.1 Bases théoriques

2.1.1 Parallaxe, stéréoscopie et photo-cohérence

Le parallaxe est un des phénomènes qui permettent à l'être humain de percevoir la profondeur des objets. Si on observe un point de l'espace – la pointe d'un crayon par exemple – avec un oeil puis l'autre, on se rend compte que le point se déplace tout dépendant de l'oeil qui le regarde. Plus l'objet est près de nous, plus ce déplacement sera important. On peut donc mesurer la distance de ce déplacement, appelé *disparité*, pour estimer la profondeur de l'objet.

C'est sur ce principe que se base la stéréoscopie classique: soit deux caméras placées côte à côte et regardant la même scène, il est possible de mettre en corres-

(a) caméra gauche (b) caméra droite

(c) carte de disparité

Figure 2.1: Exemple de stéréoscopie dense

pondance, dans les deux caméras, des points précis de la scène pour évaluer leur profondeur. Dans le type de stéréoscopie qui nous intéresse, la *stéréoscopie dense*, on tente de mettre en correspondance tous les pixels d'une image de référence par rapport à ceux de l'autre (figure 2.1). On pourra donc avoir une *carte de disparité* de la scène indiquant le déplacement associé à chaque pixel et donc sa profondeur relative.

Si le déplacement entre les caméras est horizontal et que leurs axes optiques sont parallèles, il est possible de chercher le pixel correspondant au pixel de l'image de référence sur la ligne horizontale correspondante, ligne qu'on appelle *ligne épipolaire* [37] (figure 2.2). Lorsqu'on dispose de plusieurs caméras et que les caméras sont

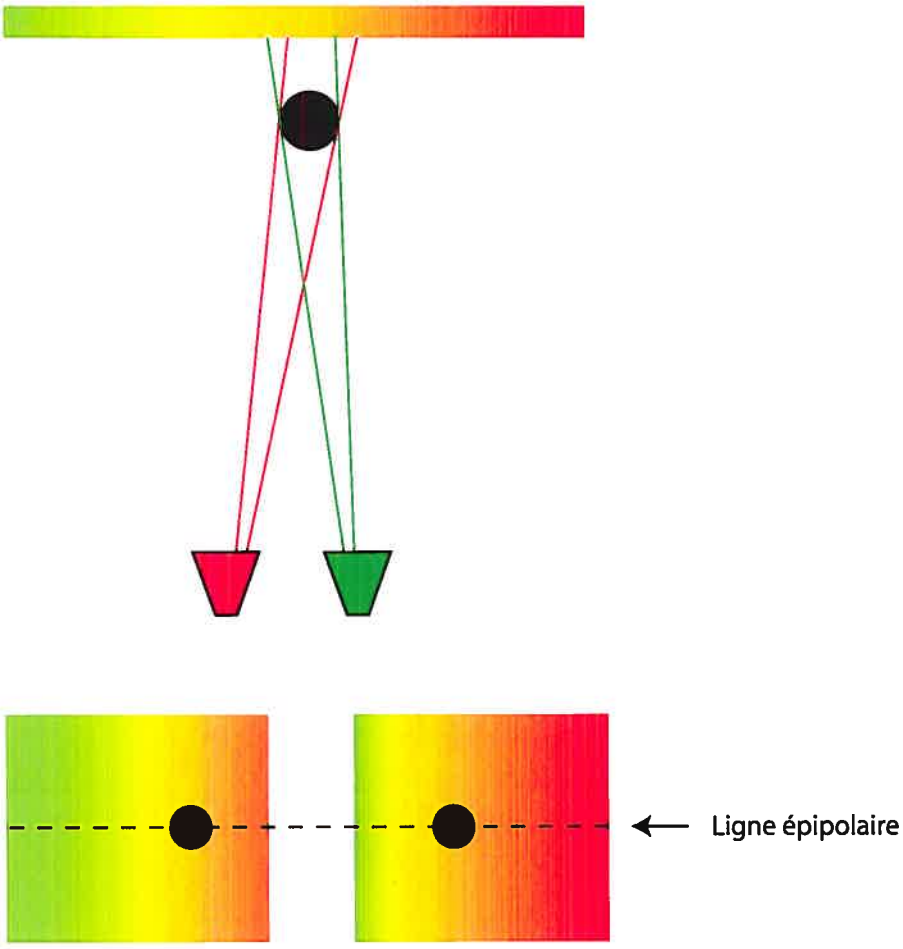


Figure 2.2: Lorsque les caméras sont côte à côte et que leurs axes optiques sont parallèles, les pixels correspondants seront placés sur une ligne horizontale appelée *ligne épipolaire*

placées de manière arbitraire, il est plus facile d'aborder le problème en sens inverse. Plutôt que de chercher une correspondance pour chaque pixel, on peut étudier un point du volume et voir si les pixels où il se projette peuvent être mis en correspondance: c'est ce qu'on appelle le principe de la *photo-cohérence*. Selon ce principe, un point de la surface apparaîtra d'une manière prévisible par chacune des caméras qui le voient. Si on connaît le modèle d'illumination de la scène, la position des caméras et les propriétés de la surface, on peut calculer la probabilité qu'un point de l'espace fasse effectivement partie de la surface en étudiant si les points correspondants sont cohérents avec notre modèle. En pratique, on pose généralement l'hypothèse que la surface est lambertienne (tel que défini à la section 1.2), et donc que les points extraits des images ne peuvent correspondre à un même point de l'espace que s'ils ont la même couleur (figure 2.3).

2.1.2 Stéréoscopie classique et reconstruction volumétrique

La reconstruction tridimensionnelle volumétrique peut être considérée comme une extension des techniques de stéréoscopie classiques. Dans le cas de la stéréoscopie classique, on cherche seulement à retrouver la profondeur des éléments d'une image – on a donc un modèle partiel de la scène, ce qu'on appelle du $2D + \frac{1}{2}$ – alors que la reconstruction volumétrique cherche à constituer un modèle 3D complet de la scène, dont la structure est définie sur tous ses côtés. Les deux méthodes se basent sur les principes de mise en correspondance des pixels décrits à la section précédente.

Cependant, les deux méthodes diffèrent dans leur traitement du phénomène de l'occlusion. Lorsque les caméras sont placées près les unes des autres et qu'elles sont orientées dans une direction semblable, comme c'est généralement le cas en stéréoscopie classique, on peut supposer qu'elles voient toutes les mêmes points de la scène. En pratique, ce n'est pas toujours le cas: certains points seront bien visibles à partir d'une caméra, mais ils seront masqués par un élément de la scène dans les autres. Ce phénomène, appelé occlusion, est cependant habituellement négligeable

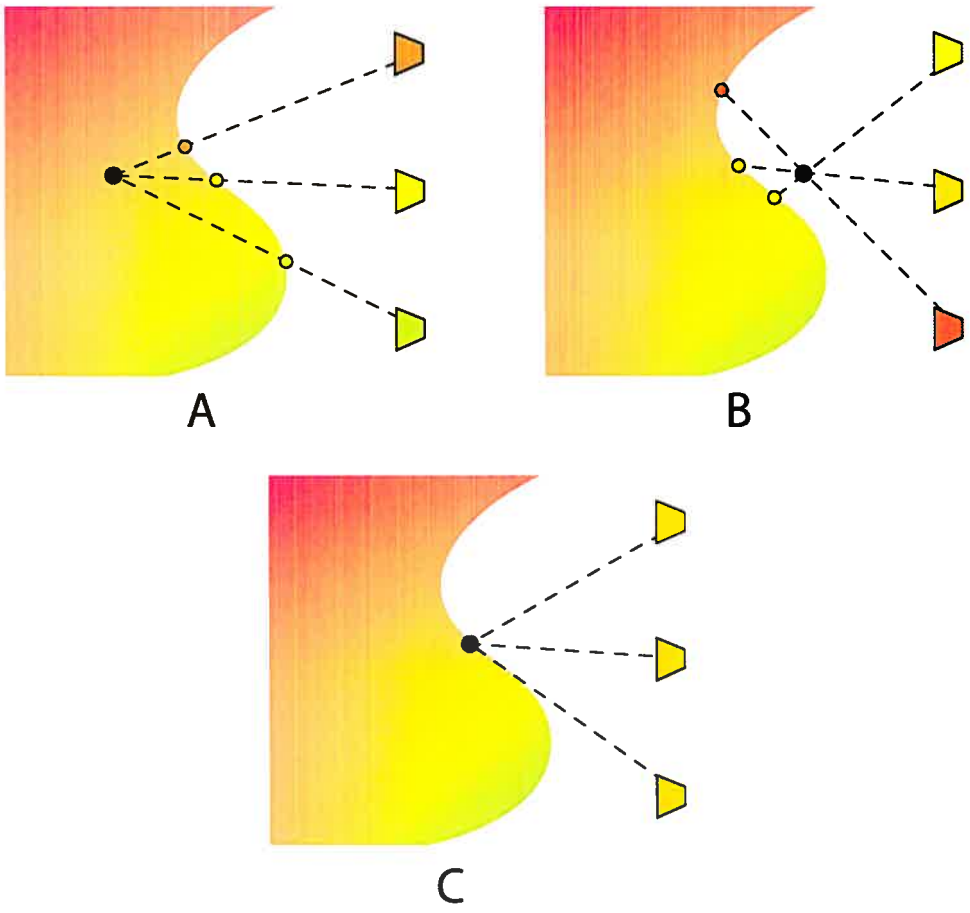


Figure 2.3: Principe de la photo-cohérence: un point situé à l'intérieur ou à l'extérieur du modèle (A et B) apparaîtra avec des couleurs différentes selon le point de vue de l'observateur, alors qu'un point placé sur la surface (C) apparaîtra avec la même couleur de toutes les caméras.

ou supposé négligeable.

Dans la reconstruction volumétrique en revanche, on ne peut plus ignorer ce phénomène [18]. Si deux caméras regardent des côtés opposés d'un sujet, elles auront des vues complètement différentes de celui-ci et il ne sera pas possible d'obtenir une correspondance directe entre les points de leurs images. Les méthodes de reconstruction volumétriques doivent donc obligatoirement tenir compte du problème de l'occlusion.

2.1.3 Algorithme du flot maximum dans un graphe

Le problème du flot maximum dans un graphe peut être défini comme suit: soit un réseau de transport d'un produit quelconque – par exemple, un système de canalisations – on cherche à déterminer le débit maximal que peut supporter ce réseau. Pour ce faire, on modélise le réseau de transport par un graphe formé de noeuds reliés arbitrairement par des arcs de capacité variable, ainsi que de deux noeuds particuliers appelés la source (étiquetée s) et le drain (étiqueté t). Dans le cas d'une canalisation, les noeuds représenteraient des embranchements, les arcs représenteraient les tuyaux eux-mêmes, la source serait le robinet d'entrée et le drain serait la sortie de la canalisation. La capacité d'un arc sera proportionnelle au débit maximal autorisé pour un morceau de tuyau, et le flot maximum représentera le débit à la sortie lorsque la canalisation a atteint son point de saturation (figure 2.4).

La saturation du réseau va se produire à certains endroits précis qui forment un goulot d'étranglement; ces segments correspondent aux arcs saturés du graphe. Une des propriétés de ces arcs, c'est qu'ils forment une coupe minimale du graphe: ils permettent de séparer le réseau en deux sections distinctes de façon à ce que les tuyaux coupés aient une capacité minimale.

Mais l'algorithme du flot maximum sert à bien plus que simuler des systèmes de tuyauterie. On peut modéliser un système de contraintes avec un graphe de façon à ce que chaque arc corresponde à un élément de solution possible. Couper le graphe

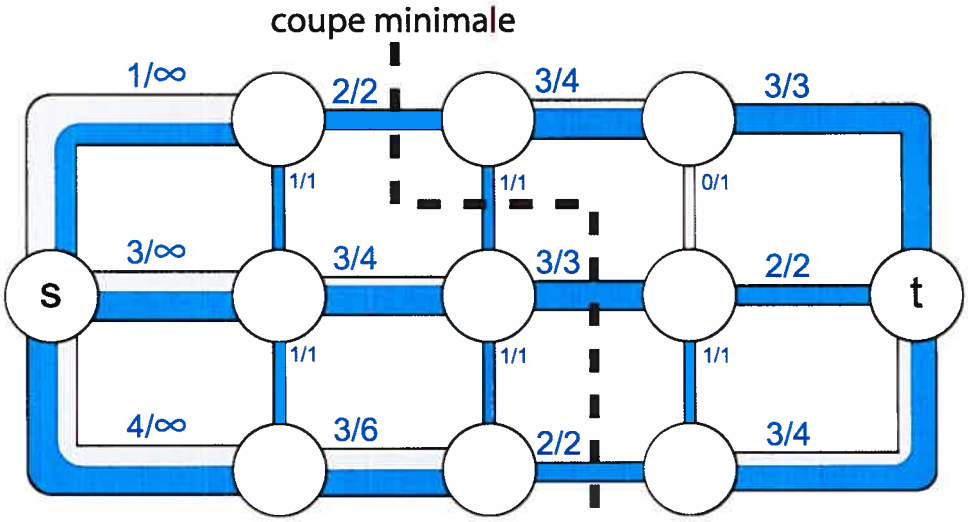


Figure 2.4: Exemple de flot maximum dans un graphe

[Exemple de flot maximum dans un graphe: on modélise ici le graphe comme un système de tuyaux dont la taille représente la capacité totale des arcs et la fraction en bleu illustre la quantité de flot qui y passe.]

à cet endroit correspondra alors à choisir cet élément de solution; il faut donc que la capacité de l'arc soit proportionnelle au coût qu'il y a à faire ce choix. À ce moment, on pourra trouver la solution ayant le coût minimal en cherchant le flot maximum dans le graphe, et en extrayant la coupe minimale résultante.

2.2 Approches majeures en reconstruction 3D

Les méthodes de reconstruction 3D se divisent en deux grandes classes: les méthodes basées sur l'intersection de silhouettes et les méthodes basées sur la photo-cohérence des pixels.

2.2.1 Reconstruction volumétrique par intersection de silhouettes

Les premiers travaux en reconstruction tridimensionnelle portaient sur la reconstruction par silhouettes. La technique consiste à segmenter, dans les images 2D, la scène à reconstruire de son arrière-plan. On forme ainsi une silhouette que l'on peut déprojeter dans l'espace pour former un volume. L'intersection de tous ces volumes constitue une enveloppe externe du sujet à reconstruire (figure 2.5). Il s'agit d'une méthode simple et rapide, qui peut être mise en oeuvre de façon très efficace [7, 36]. La méthode de base a été développée par Martin et Aggarwal [24]. Diverses innovations ont été apportées à la méthode de base, notamment l'utilisation d'*octrees* pour obtenir des algorithmes plus rapides et moins gourmands en mémoire [8].

Comme on ne cherche pas à trouver explicitement une correspondance entre les pixels des différentes images, le problème de l'occlusion ne se pose pas. En revanche, la méthode comporte des limitations importantes. Tout d'abord, le processus de segmentation de la scène nécessite habituellement un contrôle de l'environnement pour obtenir des résultats fiables, ce qui empêche le développement d'une méthode vraiment générique. De plus, la méthode de reconstruction par silhouettes ne permet pas de modéliser certains types de concavités. Il est impossible de retrouver un point de l'objet si celui-ci se trouve dans un trou de la surface. Plus formellement, on observe que si toutes les lignes tangentes à ce point intersectent la surface, il sera impossible de le reconstruire avec la méthode de reconstruction par silhouettes [24].

À cause de ces lacunes, la méthode de reconstruction par intersection de silhouettes a reçu beaucoup moins d'attention dans les dernières années, laissant la place aux méthodes de reconstruction par photo-cohérence.

2.2.2 Reconstruction volumétrique par photo-cohérence

Les techniques de reconstruction par photo-cohérence des pixels se basent sur l'hypothèse selon laquelle un point de la surface aura une couleur prévisible une fois projeté

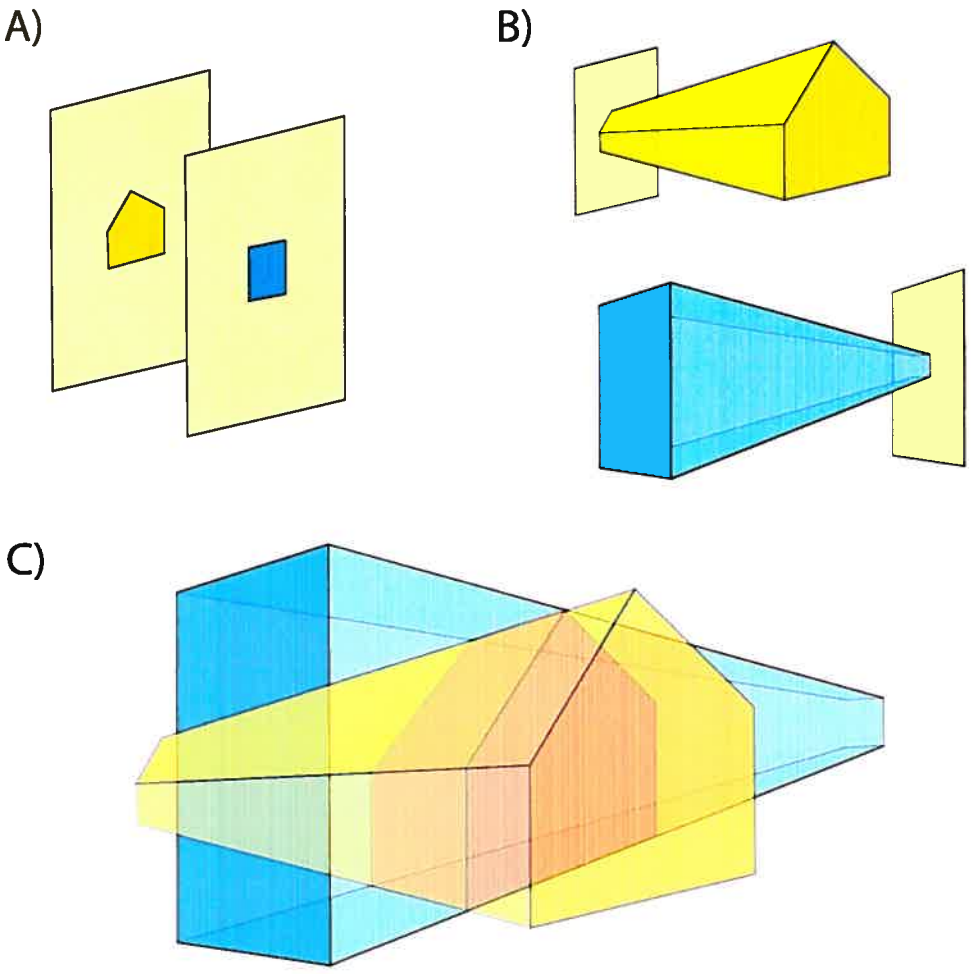


Figure 2.5: Reconstruction volumétrique par intersection de silhouettes:

A) Segmentation des silhouettes

B) Volume formé par les silhouettes projetées

C) Intersection des volumes

en 2D dans le plan de la caméra. En étudiant un ensemble de pixels, on peut donc déterminer s'il est possible ou non qu'ils correspondent à un même point de l'espace (section 2.1.1) .

Cette idée était utilisée depuis longtemps en stéréoscopie classique, mais Seitz et Dyer [33] ont été les premiers à en tirer profit pour développer une méthode de reconstruction 3D. Leur principale innovation consiste à traiter les occlusions en balayant les voxels un à un selon un ordre bien défini. Les caméras doivent être placées de manière à ce que le volume convexe formé par leurs centres ne contienne aucun élément de la scène. On peut ensuite établir un ordre de visibilité des voxels à partir de ce volume: les voxels placés plus près du volume seront vus avant ceux plus distants. On peut donc déterminer si un point de l'espace est en occlusion dans une caméra donnée en n'étudiant que l'état des voxels qui le précèdent dans l'ordre de balayage. Si c'est le cas, on peut exclure cette caméra du calcul de la photo-cohérence du voxel. Une fois que l'opacité des voxels d'une couche a été établie, on peut passer à la couche suivante.

Le *space carving* [21] vient étendre cette méthode en enlevant les restrictions sur le positionnement des caméras. La technique consiste à balayer le volume de reconstruction, initialement complètement plein, selon différents plans. À chaque étape, on cherche à retirer des voxels dont la photo-cohérence est inférieure à un certain seuil. Seules les caméras placées en deçà du plan, et dont la vue n'est pas masquée par d'autres voxels solides, sont utilisées pour mesurer la photo-cohérence d'un voxel. On peut ainsi sculpter progressivement le volume jusqu'à obtenir une reconstruction assez fidèle de la scène.

La reconstruction par *space carving* a donné lieu à plusieurs variantes apportant certaines améliorations à la technique originale. La méthode du *generalized voxel coloring* [12] propose une stratégie qui permet d'utiliser toutes les caméras qui voient un voxel, et pas seulement celles placées en deçà du plan de balayage, pour calculer sa photo-cohérence. On obtient ainsi des résultats plus fiables. Kutulakos propose la

méthode du *approximate n-view stereo* qui permet d'obtenir de bons résultats malgré la présence d'erreurs de calibrage ou de déplacements du sujet. La technique consiste à établir une correspondance entre des disques plutôt qu'entre des pixels individuels: si deux disques ont un pixel en commun, ils sont jugés équivalents. La méthode est donc beaucoup plus résistante aux erreurs. Prock et Dyer [27] présentent, quant à eux, une méthode beaucoup plus rapide de reconstruction faisant appel aux *octrees*. Une première reconstruction grossière est effectuée, puis seuls les voxels placés en périphérie de la scène sont subdivisés et raffinés. On réduit ainsi considérablement le temps de reconstruction nécessaire, ainsi que l'espace mémoire utilisé.

Toutes ces méthodes présentent cependant l'inconvénient d'être des méthodes locales. Chaque point de la scène est traité indépendamment de ses voisins. Or, on sait que les surfaces réelles sont généralement continues, et donc qu'il existe une interrelation entre l'état de deux points de l'espace voisins. Il serait intéressant de tirer profit de cette hypothèse, que l'on nomme cohérence spatiale, afin de rendre la méthode plus robuste aux différentes erreurs qui viennent fausser les reconstructions.

De plus, les processus de reconstruction 3D actuels sont séquentiels, c'est à dire les décisions prises à un moment donné vont influencer toutes les décisions prises par la suite. On ne peut pas revenir en arrière pour corriger les décisions erronées, ce qui fait qu'une petite erreur commise au début du processus pourra se répandre à toute la reconstruction. Les méthodes globales proposent une solution à ces deux problèmes en résolvant le problème simultanément pour tous les points de la scène tout en tentant compte des interrelations entre ceux-ci.

2.3 Approches globales en stéréoscopie classique

Les méthodes de reconstruction globales ont déjà fait leurs preuves dans le domaine de la stéréoscopie classique. La façon la plus simple de résoudre le problème de la stéréoscopie est de faire une recherche directe: pour chaque pixel de l'image de

référence, on cherche le pixel qui lui ressemble le plus dans une région d'intérêt de l'autre image. En pratique, il n'existe que rarement une seule mise en correspondance possible. C'est particulièrement vrai lorsque les images présentent des zones peu texturées, voire uniformes. Le problème est alors sous-déterminé. On propose donc d'ajouter une contrainte de cohérence spatiale au problème, tenant compte du fait que deux pixels voisins ont de bonnes chances d'avoir une profondeur semblable.

La stéréoscopie par fenêtres tire profit de cette cohérence en mesurant le coût de correspondance des régions qui entourent deux pixels plutôt que de mesurer le coût de correspondance des pixels eux-mêmes. On réduit donc les risques d'obtenir une fausse mise en correspondance de deux pixels non reliés. Le principal problème de cette méthode est de déterminer la taille de ces régions: de trop grandes fenêtres forceront un lissage excessif des discontinuités tandis que de trop petites fenêtres ne permettront pas de reconstruire une bonne carte de disparité dans les régions peu texturées. Il est impossible de trouver une seule taille de fenêtre optimale, et bien qu'il existe des méthodes où la taille des fenêtres est adaptative [39], celles-ci demeurent inefficaces.

La technique de programmation dynamique est une autre approche permettant de tirer profit de la contrainte de cohérence spatiale de la scène. La méthode permet de minimiser le coût total d'une solution le long des lignes épipolaires tout en encourageant les pixels voisins à avoir une disparité similaire. Cependant, aucun lissage spatial n'est effectué perpendiculairement aux lignes épipolaires, ce qui crée des effets de peigne près des discontinuités. Certains chercheurs [1, 10] ont proposé des techniques de programmation dynamique permettant d'introduire un certain lissage vertical des images, mais le lissage n'est pas uniforme dans toutes les directions et les résultats obtenus demeurent bruités.

2.3.1 Approche coopérative

L'algorithme de stéréoscopie coopératif, introduit par Marr et Poggio [23] puis repris par Zitnick et Kanade [42], peut être considéré comme la première stratégie offrant un véritable lissage 2D de la solution. Cette approche consiste à modéliser l'espace des solutions par un tableau en trois dimensions, dont chacune des cases correspond à une paire pixel-disparité possible. Chaque case est initialisée avec un score représentant la qualité de la correspondance de la paire. On définit ensuite une région de support, correspondant au voisinage en trois dimensions de chaque case. On définit aussi une région d'inhibition contenant toutes les cases placées dans le même rayon de caméra que la case étudiée.

C'est alors qu'entre en jeu la partie globale de l'algorithme: la valeur de chaque case est mise à jour en fonction de son voisinage. La fonction de mise à jour encourage les pixels voisins à avoir une disparité semblable, forçant ainsi la solution à respecter la contrainte de continuité des surfaces. Elle est aussi conçue de manière à créer une compétition entre les cases placées le long d'un même rayon. La case la plus probable verra son score augmenter, alors que les autres cases de la région d'inhibition verront leur score diminuer. La contrainte d'unicité est ainsi respectée. On itère ainsi jusqu'à convergence du système, et on choisit ensuite, pour chaque pixel, la disparité ayant le meilleur score comme valeur finale.

2.3.2 Propagation de croyances (belief propagation)

L'approche par propagation de croyances [35] est basée sur une stratégie semblable. On construit d'abord un graphe contenant un noeud par pixel. Ces noeuds peuvent prendre n états distincts, représentant chacun un niveau de disparité. On peut estimer grossièrement la probabilité que le noeud prenne un état à partir des images de référence: c'est ce qu'on appelle les observations locales. Chaque noeud fera ensuite part de ses croyances initiales à ses voisins en leur envoyant des messages. Ces

messages sont pondérés par le niveau de compatibilité des voisins: plus deux voisins sont compatibles, plus leur influence mutuelle sera importante. Les croyances sont mises à jour en mettant en commun les messages reçus et les observations locales. On itère ainsi jusqu'à convergence du système, et on choisit, pour chaque noeud, l'état que l'on croit être le plus probable.

On peut concevoir la méthode de propagation des croyances comme une technique de minimisation d'une fonction d'énergie globale, comprenant un terme mesurant à quel point la solution explique les images saisies, et un terme mesurant à quel point la solution respecte les contraintes de lissage spatial. Cette façon d'aborder le problème de la stéréoscopie comme la minimisation d'une fonction globale nous amène aux techniques de graphe, qui proposent une stratégie différente pour résoudre un problème semblable.

2.3.3 Approches basées sur les graphes

Des méthodes générales comme le recuit simulé ou la descente de gradients [30] permettent de minimiser une fonction d'énergie quelconque. Ces méthodes sont toutefois extrêmement coûteuses, et ne sont pas toujours garanties de converger vers un minimum global, ce qui les rend difficilement applicables au problème de la stéréoscopie. Certaines fonctions d'énergie peuvent cependant être minimisées en modélisant le problème par un graphe.

La technique du flot maximum [28] propose une première solution par graphe au problème de la stéréoscopie. On construit un graphe comprenant un noeud par paire pixel-disparité possible. Le graphe comprend aussi une source et un drain représentant respectivement l'avant et l'arrière de la scène. Les noeuds sont reliés à leurs voisins latéraux par des arcs de lissage ayant une capacité constante. Chacun des noeuds est aussi relié à son voisin arrière par un arc dont la capacité correspond à son coût de correspondance. Les noeuds ayant la disparité minimale sont reliés à la source et les noeuds ayant la disparité maximale sont reliés au drain. On se retrouve donc avec un

graphe de forme cubique qui présente une multitude de chemins traversant le graphe de la source vers le drain.

On peut trouver une solution optimale en forçant une certaine quantité de flot le long de ces chemins. Le flot maximum pouvant circuler dans le graphe est limité par la capacité des arcs qu'il traverse. Il y aura donc des arcs saturés, et l'ensemble de ces arcs forme ce qu'on appelle la coupe minimale du graphe: l'ensemble des arcs séparant le graphe en deux graphes distincts, et dont la capacité totale est minimale [9]. Comme ces deux graphes sont associés respectivement à l'avant et à l'arrière de la scène, on a trouvé la profondeur de la scène. La recherche de la coupe minimale se fait avec l'algorithme de préflot réétiqueter-vers-l'avant (*preflow push-relabel*) [5, 9] qui permet de trouver de façon rapide le flot maximum dans un graphe.

Boykov et Zabih [3] proposent une approche différente. Le problème posé en est un d'étiquetage: on cherche à associer une étiquette représentant un niveau de disparité à chacun des pixels de l'image de référence. Au départ, les étiquettes sont initialisées à une valeur arbitraire. On tente ensuite de modifier légèrement l'étiquetage pour améliorer progressivement la solution. Deux approches similaires sont proposées: la méthode de l'expansion alpha et la méthode de l'échange alpha-bêta. Seule la première technique sera expliquée ici. On peut, à chaque itération, donner une étiquette alpha (où alpha représente un des niveaux de disparité possible) à un nombre quelconque de pixels pour améliorer la solution totale. Pour déterminer le choix de pixels optimal, on construit un graphe modélisant la scène et on cherche la coupe minimale divisant les pixels en deux sous-graphes: les pixels gardant leur étiquette originale et les pixels auxquels on assigne l'étiquette alpha. Si ce nouvel étiquetage donne une meilleure solution que l'étiquetage précédent, on le conserve. On procède ainsi avec différentes valeurs de alpha jusqu'à ce qu'il ne reste plus de mouvement possible améliorant la situation.

Cette méthode n'est pas garantie de trouver un minimum global. En revanche, elle permet de minimiser des fonctions beaucoup plus complexes, pour lesquelles le

problème d'optimisation exacte est NP complet. On peut, entre autres, permettre un lissage plus faible entre les pixels aux disparités.

2.3.4 Conclusion sur les approches globales en stéréoscopie classique

Les approches globales ont donc fait leurs preuves en stéréoscopie. Dans leur comparaison des différentes méthodes actuelles de stéréoscopie dense [31], Scharstein et al. concluent d'ailleurs que les méthodes d'optimisation globale en deux dimensions obtiennent des performances significativement meilleures aux méthodes locales. Cette affirmation est toujours vraie, puisque selon le site du laboratoire du Middlebury Stereo Research [32], huit des dix méthodes actuelles les plus performantes en stéréoscopie sont basées sur une approche globale, que ce soit la coupure de graphe, la propagation de croyances ou l'approche coopérative.

2.4 Techniques hybrides

2.4.1 Stéréoscopie avec modélisation des occlusions

Jusqu'à maintenant, nous n'avons étudié que des méthodes de reconstruction 3D sans optimisation globale, ou des méthodes de stéréoscopie classiques avec une approche globale, mais sans gestion des occlusions. Quelques techniques ont été proposées pour joindre le meilleur de ces deux mondes. Dans un premier temps, deux équipes de chercheurs ont proposé des techniques de stéréoscopie classiques permettant de résoudre le problème de l'occlusion.

La technique du *visibility reasoning*, proposée par Kang et al. [17], consiste à résoudre itérativement la carte de disparité d'une scène avec une méthode de graphe, tout en "gelant" définitivement 15 % des meilleurs pixels à chaque itération. Les pixels ainsi fixés peuvent donc être utilisés pour déterminer la visibilité des voxels restants. Ici, la gestion des occlusions est implicite, c'est à dire qu'on détecte l'occlusion en regardant où la solution est peu précise, et on bouche les trous ainsi formés à partir

des pixels voisins. Dans le même article, les auteurs proposent des solutions locales au problème de l'occlusion, qui s'avèrent plus efficaces que la méthode globale proposée lorsque l'occlusion est faible. Il est cependant pensable que dans un contexte de reconstruction 3D, où l'occlusion est très importante, la technique globale soit valide.

Nakamura et al. [25] proposent une approche empirique de gestion des occlusions, où on applique différentes combinaisons de caméras et on choisit celle qui donne le meilleur coût avec un maximum de caméras. On ne vérifie cependant pas si cette combinaison de caméras est compatible avec la géométrie de la scène obtenue; en d'autres mots, on ne valide pas que les caméras non utilisées étaient effectivement en occlusion une fois qu'on a trouvé une carte de disparité de la scène. La méthode parvient tout de même à bien gérer l'occlusion lorsque les caméras sont placées selon une configuration bien précise – selon une matrice carrée, sur un même plan, avec leurs axes optiques perpendiculaires au plan de la scène – et qu'elles sont testées avec des configurations prédéterminées.

Kolmogorov et Zabih proposent eux-aussi une approche globale au problème de la stéréoscopie avec traitement des occlusions [18]. Le problème consiste à minimiser une fonction d'énergie comprenant un terme de données, un terme de lissage et un terme modélisant l'occlusion. Le terme d'occlusion, qui nous intéresse ici, se base sur l'hypothèse suivante: si un point $< p, l >$ fait partie de la solution, et que les pixels p et q sont voisins dans la caméra i, alors $< p, l >$ occlude tous les points du rayon passant par le pixel q dans la caméra i placés au delà de la profondeur l. De là, on peut définir le terme:

$$E_{visibility}(f) = \sum_{<p,f(p)>,<q,f(q)>\in I_{vis}} \infty \tag{2.1}$$

La minimisation est ensuite faite grâce à la méthode de l'expansion α décrite plus tôt. Le problème de cette formulation, c'est qu'il est trop coûteux de modéliser l'interaction entre les voxels de toutes les caméras. On doit donc se limiter, comme

le font les auteurs, à calculer l'interaction entre les caméras voisines. Cette approche est suffisante en stéréoscopie classique, mais dans un contexte de reconstruction volumétrique il est nécessaire de tenir compte de l'interaction entre toutes les caméras. Il est donc peu probable que cette méthode puisse être étendue à un contexte plus général.

2.4.2 Approches probabilistes

Certains chercheurs ont proposé de modifier le problème de la reconstruction par *space carving* selon une approche probabiliste. L'objectif visé par ces méthodes est d'éliminer la nécessité d'utiliser une valeur de seuil pour décider de l'état d'un voxel. La méthode de Broadhurst et al. [4] propose plutôt de donner une transparence à chaque voxel étudié en fonction de la probabilité que ce voxel soit solide. Comme dans l'approche du *space carving*, on peut ensuite évaluer l'état des voxels qui étaient précédemment en occlusion. Cette fois cependant, la visibilité n'est plus binaire, mais est plutôt calculée en fonction de la transparence des voxels occludeurs.

DeBonet et Viola [13] proposent eux-aussi une approche probabiliste pour résoudre le problème de la reconstruction volumétrique. Ceux-ci partent de l'hypothèse que le volume est entièrement transparent, et que tous les voxels contribuent uniformément à la couleur des pixels où ils se projettent. À partir de cette hypothèse de base, on construit une matrice de responsabilités R, contenant une rangée par pixel de chaque image, et une colonne par voxel. Ses cases contiendront un poids correspondant à la responsabilité de ce voxel dans la couleur du pixel étudié. Au départ, la matrice est complètement vide. Pour chaque pixel, on calcule un rayon qui traverse le volume de reconstruction, et on place une valeur constante dans les cases des voxels que ce rayon traverse.

On construit aussi une matrice C, contenant la couleur estimée de chaque voxel. Cette couleur est estimée en calculant la moyenne pondérée des pixels auxquels ce voxel contribue – c'est à dire tous ceux pour lesquels la matrice de responsabilités

R a une valeur non-nulle. On évalue la concordance de chaque voxel en mesurant la différence entre sa couleur estimée, et la couleur d'un seul des pixels auquel il contribue. On réajuste la matrice de responsabilités selon cette mesure de concordance de façon à ce que la somme des responsabilités des voxels le long d'un rayon soit 1. En d'autres mots, pour un pixel donné, on augmente la responsabilité des voxels ayant une grande concordance et on réduit celle des voxels peu concordants. On itère ainsi jusqu'à ce que les couleurs estimées et les couleurs mesurées soient identiques.

Ces deux méthodes tiennent compte de l'occlusion de manière globale plutôt que locale. On n'a donc plus le problème du choix du seuil, ni l'impossibilité de récupérer d'un voxel enlevé par erreur. On ne tient toutefois pas compte de la contrainte de continuité spatiale, ce qui fait que le système est encore sensible au bruit et aux erreurs.

2.4.3 Approches par level-sets

Faugeras et Keriven [14] proposent une méthode similaire au *space carving* faisant appel à la théorie des *level-sets* pour retrouver la géométrie d'une scène. La méthode démarre avec un volume englobant toute la scène. L'objectif est de rapprocher la surface de ce volume de la scène véritable jusqu'à obtenir une reconstruction fidèle. Pour ce faire, on déforme sa surface selon un champ de vitesse pointant vers l'intérieur du volume et dont l'amplitude dépend de la proximité de la vraie solution. La vitesse de déformation est établie en calculant, pour chaque point de la surface estimée actuelle, une valeur de corrélation entre les zones où ce point se projette dans les différentes caméras. Moins la corrélation est élevée, plus rapide est la vitesse de déformation. Le calcul de la déformation elle-même se fait grâce aux équations des *level-sets* [26].

Seules les caméras qui voient directement le point sont utilisées dans le calcul de la corrélation; comme on a une description précise de la surface courante, on peut facilement déterminer dans quelles caméras un point est en occlusion. On introduit

aussi des contraintes de tension sur la surface qui évitent une trop grande dépendance au bruit. On se trouve donc à introduire une forme de lissage.

Les résultats obtenus par l'approche des *level-sets* sont très bons. Pour que le système converge, il faut cependant que le pas d'intégration soit très petit; l'algorithme est donc très lent. De plus, la surface obtenue sera obligatoirement une fonction continue, alors que les objets réels présentent fréquemment des discontinuités. Il y aura donc un gommage excessif des arêtes.

2.4.4 Approches par graphes

Dans la dernière catégorie des approches hybrides, on retrouve les méthodes qui formulent le problème de la reconstruction 3D sous la forme d'un graphe. Ces dernières sont probablement celles qui ressemblent le plus à l'approche que nous proposons.

Dans un premier temps, Snow et al. [34] proposent une approche basée, comme la nôtre, sur le flot maximum. Ils contournent cependant le problème de l'occlusion en basant leur méthode sur la technique de reconstruction par intersection de silhouettes. Les auteurs cherchent à minimiser une fonction d'énergie composée d'un terme de données et d'un terme de lissage. Le terme de données est calculé à partir d'images segmentées de la scène, récupérées en soustrayant des images statiques du fond aux images acquises, puis en binarisant le tout. Le graphe est construit de la manière suivante: on associe un noeud à chaque voxel de la scène, et on relie chacun de ses noeuds à la source et au drain. Les arcs partant de la source ont une capacité constante k. Les arcs du noeud vers le drain, eux, ont une capacité égale à la valeur de la fonction de coût, qui est calculée en extrayant les pixels où le voxel se projette et en calculant combien d'entre eux sont à l'intérieur de la silhouette. On introduit des arcs de lissage de capacité constante entre les voxels voisins pour limiter l'effet du bruit. On résout ensuite le graphe à l'aide d'un algorithme de flot maximum dans un graphe [6]. On observe cependant que les résultats présentés ne sont que

marginalement meilleurs à ceux obtenus avec une intersection de silhouettes classique. De plus, leur approche conserve toutes les lacunes des méthodes par silhouettes, soit les problèmes de segmentation ainsi que l'impossibilité de reconstruire certains types de concavités.

Vogiatzis et al. [40] proposent eux-aussi une approche par graphe au problème de la reconstruction 3D, mais avec une fonction de coût basée sur la photo-cohérence. Leur méthode nécessite une solution initiale assez précise, fournie par l'utilisateur, dont on cherchera à raffiner la surface. On bâtit un graphe en échantillonnant la surface et en calculant la normale vers l'intérieur à ces points. On étudiera ensuite la fonction de coût à plusieurs sites, placés le long de cette normale.

On construit un graphe bayesien en associant un noeud à chaque point de la surface, n états possibles pour chacun de ses noeuds correspondant aux sites étudiés, et des arcs entre les noeuds voisins. On pourra ensuite trouver, pour chaque noeud, l'état le plus probable grâce à un algorithme de propagation des croyances. Cet algorithme, dont on a parlé plus en détail à la section 2.3.2, nécessite une mesure de la compatibilité entre deux voisins; c'est l'équivalent du terme de lissage dans l'approche par flot maximum. Cette mesure sera ici proportionnelle à la distance entre deux points voisins.

Lorsqu'on calcule la fonction de coût, on détermine, à partir de la solution fournie par l'utilisateur, quelles caméras sont en situation d'occlusion par rapport au site étudié. La méthode est donc très dépendante de cette solution initiale. De plus, si le domaine de recherche est trop grand, les sites étudiés risquent de se superposer et la solution finale pourrait présenter des auto-intersections. Les auteurs ne mentionnent pas comment cette situation est gérée, mais on peut supposer que le domaine de recherche est réduit de manière à éviter que le problème se pose. On ne peut donc reconstruire que de faibles variations de la surface originale.

Par ailleurs, les résultats présentés par les auteurs ne permettent pas de conclure à la validité de la méthode: le seul cas véritablement volumétrique présenté est un

exemple synthétique extrêmement simple, les autres résultats illustrés démontrant plutôt la performance de l'algorithme dans le cas plus simple de la stéréo classique.

2.5 Conclusion

On peut constater que les approches globales en reconstruction 3D ont été peu explorées. Les rares approches mettant en commun une recherche de solution globale avec une gestion explicite des occlusions – les approches proposées par Kolmogorov, Vogiatzis et Faugeras notamment – donnent cependant des résultats encourageants. La méthode que nous proposons, qui s'inspire des travaux de Roy et Cox pour l'optimisation, mais introduit une gestion itérative explicite des occlusions, se situe donc dans un domaine prometteur et encore peu exploré de la vision.

Chapitre 3

ALGORITHME DE RECONSTRUCTION VOLUMÉTRIQUE

Ce chapitre présente le coeur même de l'algorithme développé, qu'on peut diviser en quatre grandes sections. Nous choisissons d'abord la fonction de coût qui permet d'évaluer la photo-cohérence d'un point 3D de la scène. Nous définissons ensuite le volume de reconstruction et la structure de graphe permettant la recherche d'une solution globale. Finalement, nous présentons le mécanisme itératif de gestion des occlusions et quelques raffinements de l'algorithme de base.

3.1 Fonction de coût

3.1.1 Principe de base

Comme la plupart des techniques de reconstruction 3D modernes [12, 14, 20, 21, 33] notre approche est basée sur le principe de la photo-cohérence tel que défini par Kutulakos et Seitz [21], et décrit à la section 2.1.1.

Une première fonction de coût naïve consiste à calculer la variance des pixels où se projette le centre de chacun des voxels du volume de reconstruction. Plus la variance est faible, plus grande est la probabilité d'y trouver la surface. En utilisant la variance plutôt qu'en comparant directement la valeur des pixels, on compense pour le présence inévitable de bruit dans les images. Cette fonction de coût est cependant insuffisante parce qu'elle néglige deux autres phénomènes importants: l'occlusion et le crénelage (*aliassing*).

Le phénomène de l'occlusion vient du fait qu'une partie des caméras n'auront pas une vue directe du point de l'espace étudié. Même si le point est bel et bien sur

la surface, ces caméras verront un autre côté de l'objet et les pixels qui leur sont associés viendront augmenter la valeur de la variance. On pourrait gérer ce problème directement dans la fonction de coût, en éliminant les observations aberrantes par exemple. Pour cette première fonction de coût, nous supposons plutôt que l'occlusion est gérée par un processus externe, et que les caméras utilisées lors du calcul de la fonction de coût ont toutes une vue parfaite du point étudié.

Le second problème vient du fait qu'il n'y a pas de correspondance 1:1 entre les voxels et les pixels. Typiquement, la densité des voxels est beaucoup plus faible que celle des pixels, puisqu'il faut beaucoup plus d'espace et de temps de calcul pour traiter un volume 3D qu'une image 2D de même dimension. On peut donc supposer qu'un voxel se projettera sur plusieurs pixels voisins sur une image.

Si on ne considère que le pixel central – celui correspondant à la projection du centre du voxel – dans le calcul de la fonction de coût, on n'utilise qu'une fraction de l'information disponible. Pour obtenir une fonction de coût fiable, on doit projeter le voxel au complet et tenir compte de tous les pixels correspondants.

Différentes approches ont été proposées par les chercheurs pour réaliser cette opération. Quelques-uns [12, 34] choisissent de projeter explicitement le voxel et d'extraire tous les pixels correspondants. Cette opération est cependant très coûteuse, surtout si le volume de reconstruction est adaptatif, ce qui exclut les possibilités de précalcul. Une autre approche possible [21, 27] consiste à utiliser les capacités de *texture mapping* de la carte graphique pour projeter tout un plan du volume d'un seul coup. Cette approche suppose cependant une organisation et un ordre de traitement particuliers du volume de reconstruction, contraintes qui ne sont pas respectées dans notre approche. Finalement, une dernière approche consiste à approximer la projection du voxel, soit par un masque rectangulaire [33] ou par un disque de rayon constant [20]. Nous avons choisi cette dernière approche pour sa simplicité et son efficacité.

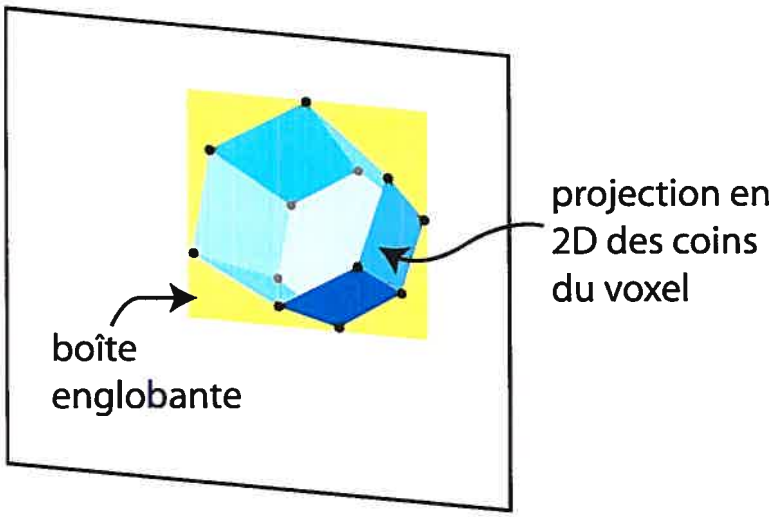


Figure 3.1: Estimation de la projection avec un masque rectangulaire englobant les sommets du voxel

Notre technique consiste à projeter les coins du voxel [1], et à trouver une boîte rectangulaire alignée sur les axes de l'image qui couvre minimalement les pixels identifiés (figure 3.1). Un des avantages de cette méthode, c'est qu'on peut extraire la moyenne et la variance d'un bloc de pixels en temps constant; on utilise pour cela une simple extension de la méthode de Crow [11, 15]. La projection des voxels ne sera donc plus dépendante de leur taille. La moyenne des pixels ainsi extraits sera ensuite utilisée pour le calcul de la fonction de coût.

[1] Il est important de noter le terme voxel réfère simplement à un élément de volume; bien que la plupart des voxels soient cubiques, les nôtres ont une structure géométrique un peu plus complexe qui sera décrite à la section 3.2.3.

La fonction de coût proposée est donc:

$$c(v) = \frac{\sum_{i=0}^{N-1}(x(v,i) - \bar{x})^2}{N} \tag{3.1}$$

où N est le nombre de caméras

$x(v,i)$ est la couleur moyenne des pixels où le voxel v se projette dans la caméra i

$\bar{x} = \frac{\sum_{i=0}^{N-1} x(v,i)}{N}$ est la couleur moyenne de tous les échantillons

3.1.2 Fonction de coût robuste

Nous proposons aussi une seconde fonction de coût, celle-là plus robuste, qui fonctionne bien même en présence d'observations aberrantes (*outliers*). En effet, notre processus de gestion des occlusions est itératif, et lors des premières itérations toutes les caméras seront utilisées. Il y a aura donc un grand nombre de points aberrants qui viendront corrompre la fonction de coût.

Un cas problématique typique est présenté en deux dimensions à la figure 3.2. On cherche à évaluer la fonction de coût aux deux points indiqués sur l'illustration. La projection du point A dans les quatre caméras donne une variance plutôt faible puisque les pixels sont tous plus ou moins dans la même gamme de couleurs, et ce même si les caméras sont toutes en situation d'occlusion. Au point B, en revanche, la variance est très élevée même si la surface passe réellement par ce point, parce que les caméras qui sont en occlusion voient un côté complètement différent de l'objet.

On voudrait donc un système qui favorise le second cas (quelques couleurs très proches, avec possiblement des observations aberrantes) plutôt que le premier (beaucoup de couleurs moyennement proches). On propose donc la fonction de coût suivante:

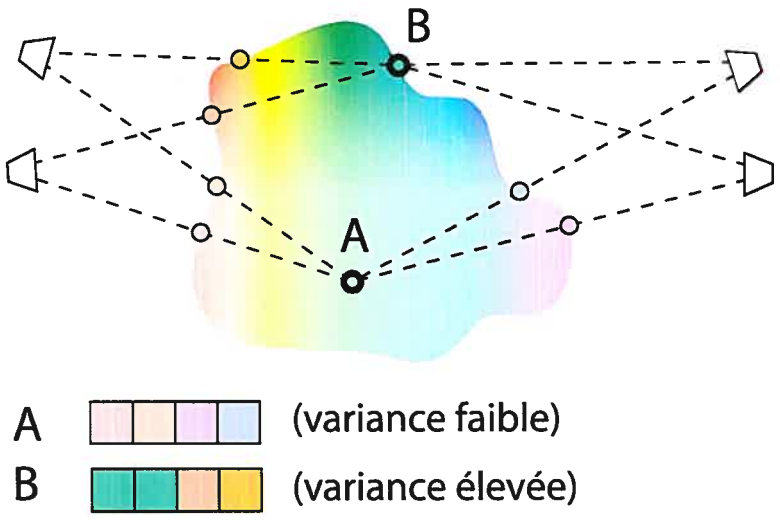


Figure 3.2: Cas problématique pour la fonction de coût de base

$$c(v) = min_{ij,i\neq j} |d(v,i) - d(v,j)| \tag{3.2}$$

où $c(v)$ est le coût du voxel v et

$d(v,x)$ est la couleur moyenne du voxel v projeté dans la caméra x.

La fonction est basée sur le principe suivant: on trouve la paire de caméras ayant la meilleure mise en correspondance, et on utilise seulement ces deux caméras pour évaluer la fonction de coût. Cette approche a l'avantage d'éliminer les caméras qui donnent des résultats aberrants, et de favoriser une mise en correspondance très élevée d'un petit nombre de caméras par rapport à une mise en correspondance moyenne de toutes les caméras. Comme le premier cas est plus probable lorsqu'il y a beaucoup d'occlusion, on peut prédire que cette fonction de coût donnera de meilleures performances que la fonction simple de l'équation 3.1.

En revanche, elle élimine de l'information qui pourrait être utile. De plus, rien ne garantit que les deux caméras choisies sont effectivement des caméras valides; il

est possible que deux points non reliés aient une couleur pratiquement identiques, et viennent fausser la fonction de coût.

Nous avons établi empiriquement que la fonction de coût robuste donne des résultats légèrement meilleurs. Comme elle est plus ambigüe que la première, elle donne un peu plus de marge de manoeuvre à l'algorithme de recherche globale et permet de donner plus d'influence au lissage. Il est toutefois important de noter que notre algorithme de reconstruction est relativement indépendant de la fonction de coût choisie. D'autres fonctions de coût plus complexes tenant compte, par exemple, de toute la gamme des couleurs de la zone où se projette un pixel [2, 20] pourraient facilement être intégrées au système.

3.2 Recherche d'une solution avec l'algorithme du flot maximum

3.2.1 Stéréoscopie classique avec l'algorithme du flot maximum

Avant d'aller plus loin dans la présentation de notre approche, il est essentiel de détailler la méthode de stéréoscopie avec flot maximum développée par Roy et Cox [28] dont notre approche s'est inspirée.

La méthode consiste à utiliser l'algorithme de flot maximum décrit à la section 2.1.3 pour minimiser une fonction d'énergie modélisant le problème de la stéréoscopie. On pose l'hypothèse que toutes les caméras sont placées d'un même côté du volume de reconstruction; le problème est alors de trouver la profondeur (ou disparité) de chaque point d'une image de référence. La stratégie consiste alors à modéliser le problème sous la forme d'un graphe dont les arcs seront associés aux couples pixel-disparité possibles, et dont la coupe minimale correspondra à la carte de disparité optimale.

Le volume de reconstruction proposé par Roy et Cox (figure 3.3) est un prisme rectangulaire faisant face à la caméra de référence. À chaque "tranche" du volume est associée une couche du graphe, comprenant un noeud par pixel de l'image. Pour

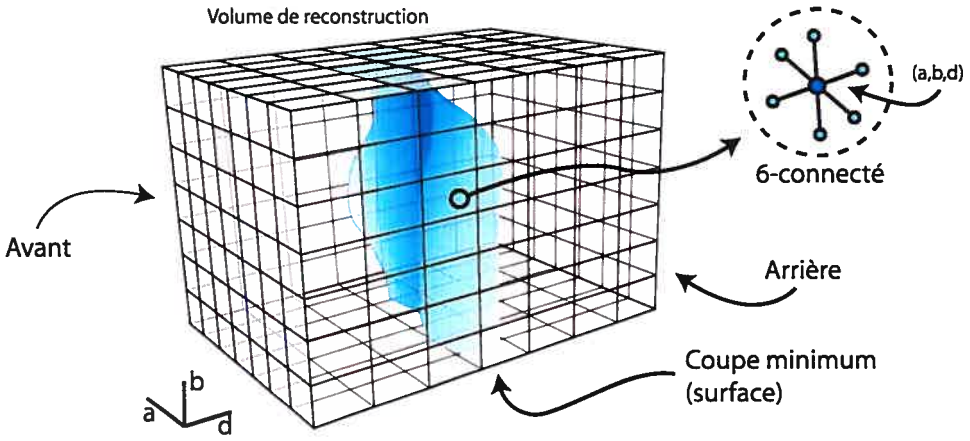


Figure 3.3: Volume de reconstruction proposé par Roy et Cox

une image de référence de $a \times b$ pixels, et d disparités possibles, on a donc $a \times b \times d$ noeuds. Des *arcs de lissage* relient les noeuds voisins d'une même couche, et des *arcs de correspondance* relient les couches entre elles. On introduit aussi deux noeuds particuliers, la source et le drain, qui seront reliés à tous les noeuds de la première et de la dernière tranche respectivement. La capacité des arcs de correspondance est directement reliée au coût du noeud d'où l'arc provient. La capacité des arcs de lissage, elle, est constante partout dans le graphe.

Une fois le graphe bâti, on peut introduire du flot à la source, et augmenter celui-ci jusqu'à ce que le graphe soit saturé. Dans le cas simple où les arcs de lissage ont une capacité nulle, le flot saturera au noeud ayant le coût le plus bas pour un pixel donné. L'algorithme se réduit donc à une simple recherche directe de la meilleure solution locale. En revanche, si les arcs de lissage ont une capacité non-nulle, le flot en excédent pourra être déversé dans les chemins voisins, à condition que ceux-ci ne soient pas eux-mêmes saturés. Cette répartition de l'excédent permet d'introduire une composante globale dans l'algorithme: la disparité à un pixel de l'image ne dépend plus uniquement de ce pixel, mais aussi de ses voisins. Si la fonction de coût est clairement minimale à une disparité précise, c'est cette disparité qui sera choisie.

Mais si la fonction de coût est ambigüe – on peut avoir, par exemple, à décider entre deux minima locaux ayant un coût très proche – le transfert de flot fera en sorte que la solution s'inspire des pixels voisins.

La quantité de lissage dépendra de la capacité des arcs de lissage: plus celle-ci est élevée, plus les discontinuités de la surface seront pénalisées. Dans le cas extrême où la capacité des arcs de lissage est infinie, l'algorithme trouvera une solution où tous les pixels de l'image ont la même disparité, correspondant à la disparité moyenne de la scène.

3.2.2 Variante volumétrique du graphe

L'objectif de notre approche est de proposer un graphe similaire à celui utilisé en stéréo, c'est à dire un graphe dont les arcs saturés par l'algorithme de flot maximum correspondent à la surface la plus probable. Il est cependant impossible de définir cette fois un volume ayant un plan avant et un plan arrière, puisqu'on cherche à reconstruire tous les côtés de l'objet. Nous devons donc repenser la forme du graphe et la circulation du flot.

L'approche proposée consiste à transformer le graphe cubique en sphère de façon à ce que son plan avant corresponde à la surface de la sphère et son plan arrière à son coeur (figure 3.4). En pratique, on propose une surface sphérique englobant tout l'objet, et on relie la source à tous les points de cette surface. On aligne la sphère par rapport à l'objet à reconstruire de manière à ce que son centre soit à l'intérieur de l'objet, et on relie le drain à ce centre. Les arcs de correspondance seront positionnés radialement pour que le flot circule de l'extérieur vers l'intérieur, et que le graphe sature là où ce flot traverse la surface de l'objet.

On bâtit le volume de reconstruction à partir d'un maillage qui approxime la forme d'une sphère échantillonnée de manière quasi-uniforme. Ce maillage, appelé *maillage patron*, sera copié en plusieurs exemplaires de tailles différentes qui seront emboîtés comme des poupées russes pour former les couches du volume (figure 3.5).

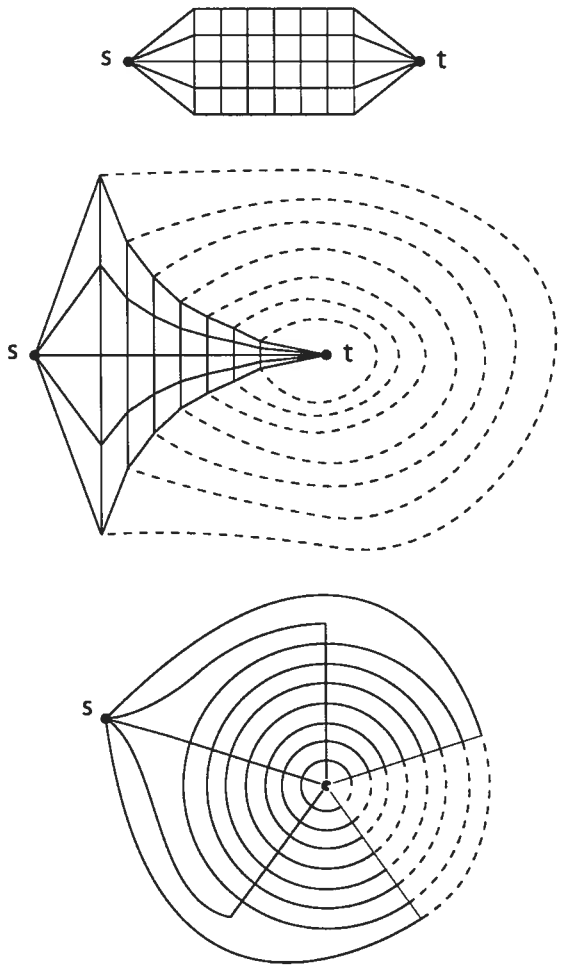


Figure 3.4: Exemple en deux dimensions de la transformation du graphe cubique en sphère

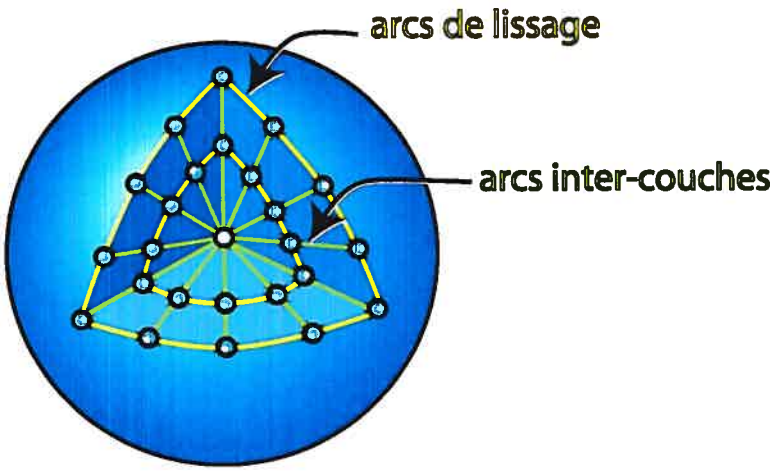


Figure 3.5: Volume de reconstruction sphérique

Plus le maillage est précis et plus le nombre de couches est élevé, plus la résolution de la reconstruction sera élevée.

On choisit le facteur d'échelle des maillages de façon à ce que la distance radiale entre chacune des couches soit uniforme. Puisque ces couches sont formées de maillages topologiquement identiques, on peut trouver une correspondance directe entre les sommets de chaque couche et établir un chemin radial de la surface extérieure vers le coeur de l'objet. Finalement, on forme un voxel autour de chaque sommet de chaque maillage, et on associe un noeud du graphe à ce voxel.

On a donc un ordonnancement semblable à celui utilisé en stéréo, où les pixels sont maintenant des sommets et où l'étiquette de disparité est maintenant le numéro de la couche. On peut aussi établir des relations de voisinage entre les noeuds d'une même couche en suivant les arêtes du maillage. Comme le maillage utilisé n'est pas un polygone régulier – puisqu'il n'existe pas de polygone régulier de plus de vingt faces, ce qui serait évidemment une résolution insuffisante pour notre problème – le nombre de voisins variera d'un noeud à l'autre. Cependant, comme le graphe demeure bien formé, ceci n'empêche pas l'algorithme du flot maximum de trouver une solution.

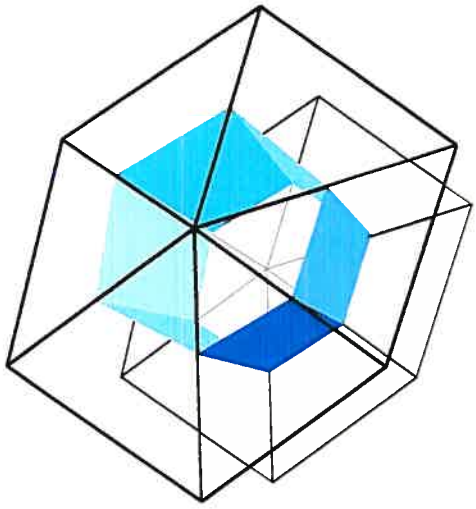


Figure 3.6: Formation d'un voxel

3.2.3 Formation des voxels

Pour former le voxel associé à un sommet du volume, on trouve le point milieu de chacune des arêtes reliées à ce volume. On observe aussi le sommet "enfant" de ce sommet, c'est à dire le sommet équivalent sur la couche intérieure suivante, et on trouve les points milieu de ses arêtes. Le voxel est formé du volume englobant de ces points milieu (figure 3.6).

3.2.4 Capacité des arcs

La capacité des arcs inter-couches est, comme en stéréo, proportionnelle au coût des noeuds. Soit un voxel v_a associé au noeud n_a. Comme ce voxel est en fait placé entre les deux couches du graphe, il est logique d'utiliser le coût de ce voxel pour calculer la capacité de l'arc reliant n_a à son enfant.

$$cap(a_{inter}) = c(v_a) \tag{3.3}$$

où $cap(a_{inter})$ est la capacité des arcs inter-couches et

$c(v_a)$ est le coût du voxel v_a

La capacité des arcs de lissage est plus complexe. Tout d'abord, la densité des noeuds par couche n'est pas uniforme, puisqu'on utilise le même nombre de noeuds par couche peu importe la taille du maillage. Les voxels des couches extérieures seront donc plus espacés que ceux des couches intérieures. Comme le lissage sert à modéliser l'interdépendance entre deux points voisins, et qu'il est logique de penser que plus ces points sont éloignés, moins ils sont interdépendants, on tiendra compte de la distance entre les voxels dans la fonction de coût. Plus précisément, on choisit de multiplier la capacité de l'arc par un facteur $\frac{1}{d}$, ce qui donne empiriquement de bons résultats .

Mais plus important encore, la capacité de base des arcs de lissage n'a plus à être constante dans notre système. **On sait qu'un arc saturé signifie que la surface coupe cet arc**. En stéréoscopie, on supposait que les caméras étaient placées sur un même plan, et rapprochées les unes des autres, ce qui fait qu'on n'observait jamais les surfaces coupant les arcs de lissage. Ce n'est cependant plus le cas dans notre approche: **les morceaux de surface associés aux arcs de lissage saturés sont aussi visibles que ceux associés aux arcs inter-couches**. La figure 3.7 illustre ce phénomène. Il est donc logique de leur associer, eux aussi, un coût qui dépend de la photo-cohérence de ce bout de surface. On estime ce coût en calculant la moyenne des coûts des voxels aux extrémités de l'arc.

La capacité de base des arcs de lissage est donc:

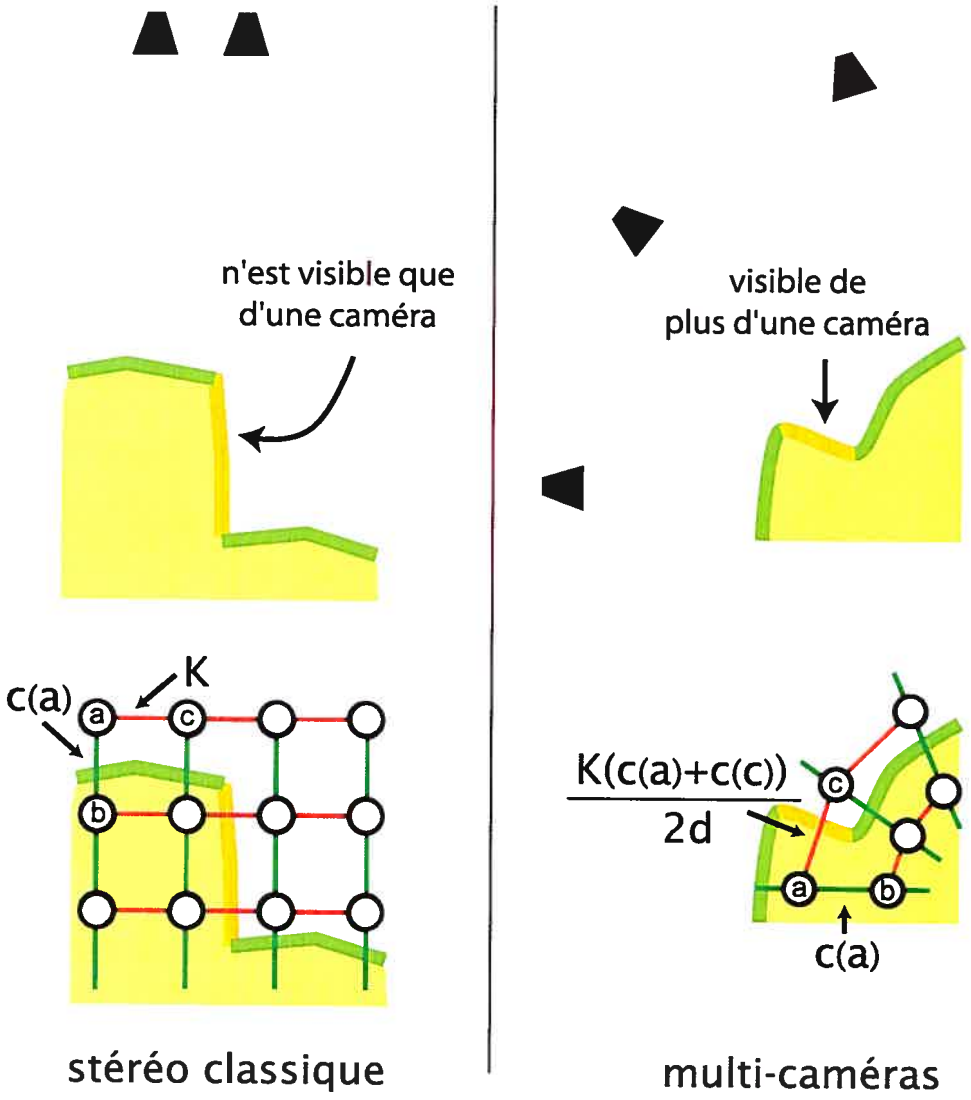


Figure 3.7: Capacité des arcs de lissage pour la stéréoscopie classique et la reconstruction volumétrique

$$cap(a_l) = K \cdot \frac{c(v_a) + c(v_b)}{2} \tag{3.4}$$

où $c(v_i)$ est le coût du voxel i et

v_a et v_b sont les voxels aux extrémités de l'arc de lissage a_l

Si on lui ajoute le facteur de distance, la capacité finale devient:

$$cap(a_l) = \frac{K}{2 \cdot d}(c(v_a) + c(v_b)) \tag{3.5}$$

3.2.5 Calcul du flot maximum

Pour la résolution de l'algorithme du flot maximum lui-même, on fait appel au code de Goldberg basé sur l'article publié avec Cherkassky [6] et distribué publiquement [16]. L'algorithme de Goldberg est basé sur le principe du préflot réétiqueter-vers-l'avant (*preflow push-relabel*) décrit dans Cormen et al. [9]. Les auteurs proposent plusieurs améliorations à cet algorithme. La première de ces améliorations, le réétiquetage global (*global relabeling*), consiste à calculer le plus court chemin du drain vers chacun des noeuds et à associer cette valeur comme étiquette initiale de hauteur des noeuds. La seconde amélioration consiste à repérer des trous dans le graphe. Si à un moment donné, aucun noeud n'a été étiqueté avec la valeur g, alors tous les noeuds dont la valeur est supérieure à g peuvent être montés jusqu'à une altitude de n. Les auteurs décrivent aussi deux façons de gérer les noeuds actifs: par FIFO, ou en ordre d'altitude du plus haut vers le plus bas (HL ou *highest label*). La combinaison de l'algorithme HL avec les améliorations de réétiquetage donnent de bonnes performances sur la plupart des problèmes.

Il est important de mentionner que tous les algorithmes de calcul du flot maximum donneront le même résultat pour un graphe donné. Cependant, les temps de calcul diffèrent grandement d'une approche à l'autre, et comme notre graphe a une taille

importante, il est crucial de choisir un algorithme efficace comme l'algorithme de Goldberg.

Une fois qu'on a calculé le flot maximum dans le graphe et trouvé la capacité résiduelle de chacun des arcs, on peut rechercher la coupe minimale – le "goulot d'étranglement" du graphe – qui correspond à la solution de notre problème. On calcule cette coupe en cherchant tous les noeuds accessibles à partir du drain par des arcs encore non-saturés. Cette recherche se fait par une simple recherche en largeur à partir du drain. Tous les noeuds ainsi accessibles seront associés au drain, et les arcs les reliant aux autres noeuds du graphe formeront la coupe. Il suffit ensuite de construire la solution en repérant les arcs inter-couches faisant partie de la coupe, et de positionner les sommets d'un nouveau maillage au point milieu de ces arcs.

3.3 Gestion des occlusions

3.3.1 Principe général

Les occlusions sont un problème inévitable de la reconstruction volumétrique. Comme l'objet à reconstruire est tridimensionnel et que les caméras sont placées tout autour de lui, il est impossible qu'une caméra voie toute la surface de l'objet.

La fonction de coût proposée à la section 3.1.2 permettra, dans une certaine mesure, de contourner le problème de l'occlusion. Mais ce n'est cependant pas suffisant pour obtenir une reconstruction fidèle; il faudra identifier les caméras en occlusion et les éliminer du calcul de la fonction de coût.

La méthode classique pour résoudre le problème des occlusions est une approche conservatrice: on n'utilise que les caméras qui sont garanties de voir les voxels exposés, et au fur et à mesure qu'on prend la décision d'éliminer un voxel visible, on expose de nouveaux voxels qui pourront à leur tour être évalués. C'est l'approche qui est généralement utilisée dans les méthodes de reconstruction 3D par photo-cohérence [12, 21, 33]. Le problème de cette approche, dont nous avons discuté à la section

2.2.2, c'est qu'elle nécessite un processus séquentiel. Il est impossible de revenir sur une décision erronnée, ou d'attendre de connaître l'état d'un voxel encore caché avant de statuer sur l'état d'un autre.

Nous proposons plutôt une approche itérative où les occlusions seront gérées progressivement. Au départ, on suppose que toutes les caméras ont une vue parfaite de toutes les surfaces, et on calcule une première reconstruction. Cette reconstruction sera évidemment inexacte puisque notre hypothèse de départ est erronée. Nous avons cependant établi empiriquement que cette reconstruction est suffisante pour prendre des décisions quant à la visibilité de la scène.

3.3.2 Mesure de l'occlusion

À partir de cette première solution, on évalue la visibilité de chacun des voxels pour toutes les caméras.

Cette étape demande une mesure de l'occlusion, qu'on peut évaluer en faisant un rendu de la scène. Considérons un pixel $p(v,i)$ correspondant à la projection du centre du voxel v dans la caméra i. On fait un rendu de la solution courante à partir de la caméra i, et on extrait la valeur du tampon de profondeur au pixel $p(v,i)$ [15]. Cette valeur est convertie au système de coordonnées de la caméra i, et nommée $z_{solution}$. On évalue aussi la profondeur du voxel v dans le même système de coordonnées, z_{voxel}.

Si $z_{solution} < z_{voxel}$, c'est que la surface de la reconstruction initiale cache le voxel v à la caméra i (figure 3.8, points B et C). On mesure cette occlusion avec

$$o(v,i) = |z_{solution} - z_{voxel}| \tag{3.6}$$

où $o(v,i)$ est l'occlusion de v dans la caméra i.

Si $z_{solution} \geq z_{voxel}$, il n'y a pas d'occlusion puisque la solution courante est placée derrière le point de l'espace étudié (figure 3.8, point A).

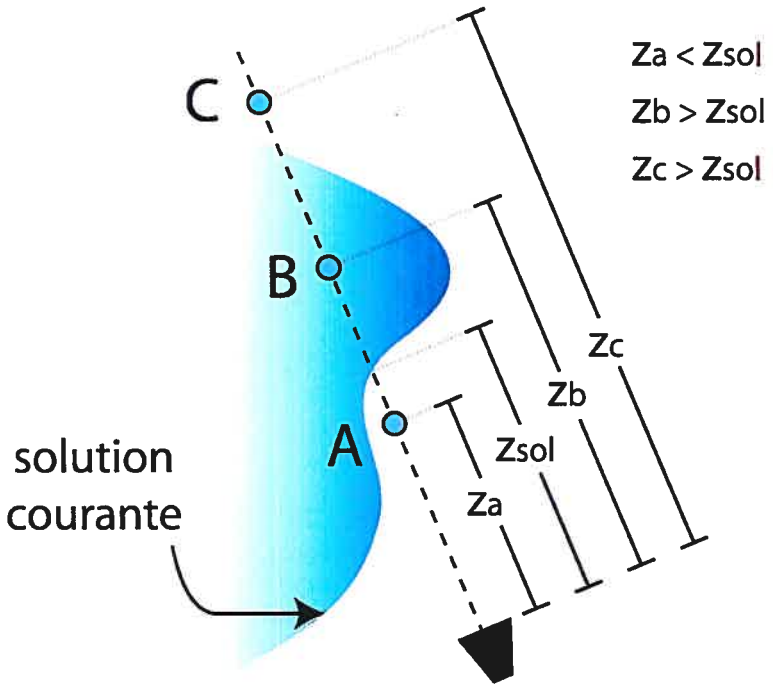


Figure 3.8: Mesure de l'occlusion: À partir de cette caméra, les voxels B et C sont en occlusion, et le voxel C est celui ayant la mesure d'occlusion la plus élevée.

3.3.3 Élimination des caméras

Une fois qu'on a obtenu une mesure de l'occlusion pour toutes les caméras par rapport à un voxel, on peut éliminer celles en occlusion. On serait tenté d'éliminer toutes les caméras qui se trouvent en occlusion. Cependant, notre solution initiale n'est pas assez fiable pour garantir qu'on ne coupera pas des caméras essentielles.

On propose plutôt une approche conservatrice qui consiste à ne couper que la caméra qui présente le plus haut niveau d'occlusion à chaque étape. Une fois une caméra coupée, celle-ci ne sera plus jamais utilisée pour ce voxel, ce qui permet à l'algorithme de converger. Comme on n'ajoute jamais de caméra, il est impossible que l'algorithme entre dans un cycle. Il est évidemment possible, même avec une approche conservatrice, qu'une caméra utile soit effectivement coupée. Mais puisque deux bonnes caméras sont suffisantes pour évaluer le coût d'un voxel, on n'utiliser qu'un sous-ensemble de toutes les caméras valides en ne perdant qu'un peu de précision.

3.4 Synthèse de l'algorithme de base

Tous les éléments de solution présentés plus tôt consistuent le coeur de notre algorithme (voir algorithme 1). À cette approche de base s'ajouteront divers raffinements qui seront expliqués plus loin. La figure 3.9 illustre les étapes principales de l'algorithme. On évalue d'abord la fonction de coût avec toutes les caméras, puis on construit une première reconstruction avec l'algorithme du flot maximum dans un graphe. Cette reconstruction servira à évaluer la visibilité et à éliminer la pire caméra. On peut alors faire une nouvelle reconstruction plus précise et itérer ainsi jusqu'à ce que les résultats soient satisfaisants ou qu'il ne soit plus possible d'éliminer des caméras.

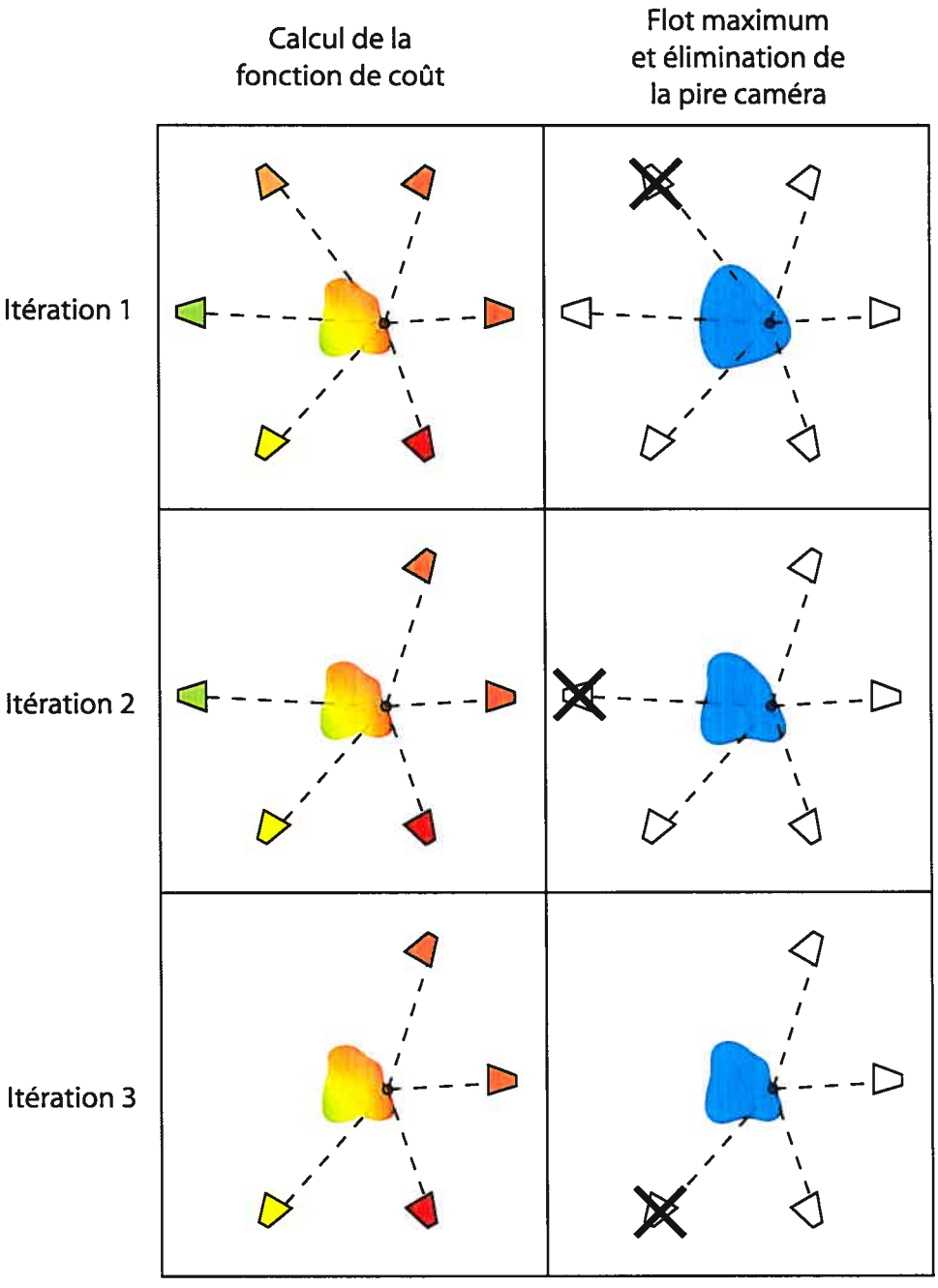


Figure 3.9: Illustration de l'algorithme de base

Algorithme 1: Reconstruction volumétrique de base avec traitement des occlusions

(1) construire le volume de reconstruction à la résolution choisie
(2) **pour** chaque voxel
(3) calculer la fonction de coût
(4) reconstruire avec l'algorithme du flot maximum
(5) **pour** chaque caméra
(6) faire un rendu du tampon de profondeur avec la solution initiale
(7) **pour** i = 0...N
(8) **pour** chaque voxel
(9) **pour** chaque caméra restante
(10) évaluer z_{voxel} et $z_{solution}$
(11) $o(v, c) = max(0, z_{voxel} - z_{solution})$
(12) éliminer, pour le voxel v, la caméra ayant la valeur d'occlusion la plus élevée
(13) calculer la fonction de coût avec les caméras restantes
(14) reconstruire avec l'algorithme du flot maximum
(15) **pour** chaque caméra
(16) faire un rendu du tampon de profondeur avec la solution courante
(17) afficher le résultat final

3.5 Raffinements de l'algorithme

3.5.1 Prétraitement des caméras

Un premier raffinement que l'on peut apporter à l'algorithme présenté, c'est l'élimination dès le départ de caméras invalides. Selon la contrainte de topologie radiale posée à la section 1.2, on peut établir que les morceaux de surface auront une orientation limitée. Il est donc possible d'éliminer les caméras qui sont pratiquement garanties d'être derrière la surface, et donc d'être en occlusion peu importe leur angle de vue. On met en oeuvre ce concept en délimitant un hémisphère centré sur le voxel en dehors duquel les caméras seront éliminées d'office. Il s'agit d'une simplification un peu abusive – on peut imaginer des cas extrêmes où les caméras hors de l'hémisphère verraient correctement la surface alors que les caméras placées directement devant le voxel n'en auraient qu'une vue imprécise – mais cela s'avère une heuristique généralement acceptable pour réduire le temps de calcul de l'algorithme.

Si on considère le vecteur $\mathbf{v}$ allant de l'origine du volume de reconstruction jusqu'au voxel étudié, et le vecteur $\mathbf{c}$ allant de l'origine vers le centre optique de la caméra. Si l'angle entre ces deux vecteurs est supérieur à 90^o, la caméra est placée dans l'hémisphère "invalide" et peut automatiquement être coupée (figure 3.10).

Ce prétraitement permettra d'éliminer dès le départ plus ou moins la moitié des caméras (si on suppose une distribution à peu près uniforme des caméras autour du modèle), ce qui améliorera l'itération initiale et accélèrera la convergence.

3.5.2 Approche à résolution multiples

Comme nous l'avons expliqué à la section 3.2.2, la résolution de la reconstruction dépend du nombre de sommets du maillage patron, ainsi que du nombre de couches. Comme les premières reconstructions ne servent qu'à évaluer l'occlusion, et seront de toute manière très grossières, il est inutile d'utiliser un graphe à la résolution maximale. On propose donc un système multi-résolutions où l'on roule le système en

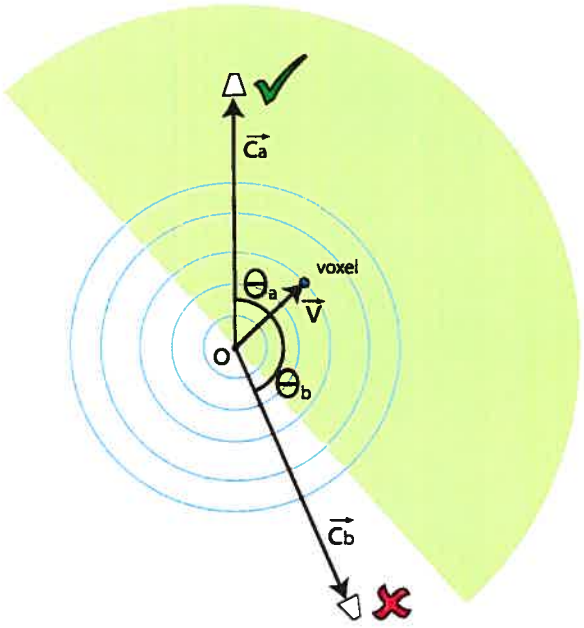


Figure 3.10: Orientations autorisées pour un morceau de surface, basées sur la contrainte radiale

basse résolution jusqu'à convergence de la visibilité, puis on utilise le volume obtenu lors de cette première phase pour évaluer la visibilité du graphe haute résolution, qu'on n'a qu'à résoudre une seule fois par la suite.

3.6 Options explorées, mais non retenues

3.6.1 Subdivision adaptative de la surface

Une des approches que nous avons explorées est la subdivision adaptative de la surface. Le principe était le suivant: à partir d'une reconstruction à basse résolution, il serait possible d'identifier les zones contenant plus de détails, et de subdiviser localement le graphe à ces endroits.

Le processus de subdivision proposé consistait à identifier des points critiques du modèle reconstruit, à subdiviser dans le maillage patron les triangles voisins de ces points critiques, puis à rebâtir un nouveau graphe avec le patron subdivisé.

Pour éviter la présence de sommets en T, un processus de subdivision de maillage en deux étapes inspiré de Krivanek et al. [19] a été suggéré: dans un premier temps on subdivise tous les triangles nécessaires, puis on balaye le graphe à la recherche de sommets en T. Les triangles touchant à de tels sommets sont découpés en deux ou trois sous-triangles pour assurer l'intégrité du modèle sans introduire de nouveaux sommets en T (figure 3.11).

Le point faible de cette approche s'est avéré l'identification des points critiques. La méthode proposée était de mesurer le rayon de courbure de la surface, et de subdiviser le volume là où la courbure était élevée. Cette méthode était efficace pour raffiner des discontinuités majeures de la surface, comme l'arête d'une demi-sphère par exemple. En revanche, elle ignorait complètement les détails de haute fréquence qui n'étaient pas détectables dans la reconstruction basse résolution. Nous avons donc opté pour une approche qui subdivise uniformément le modèle.

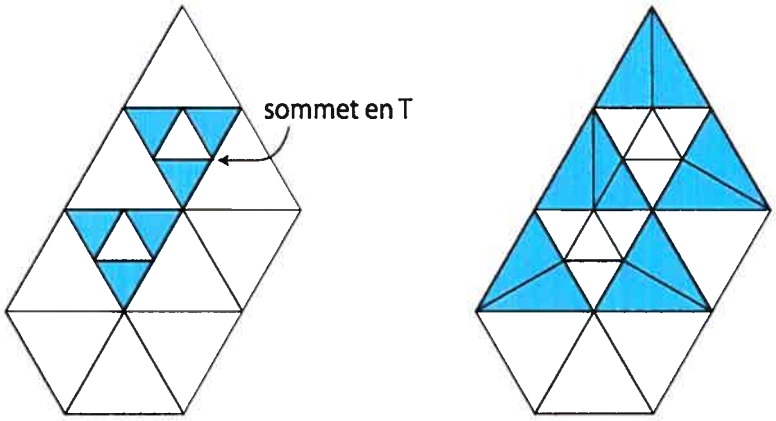


Figure 3.11: Subdivision adaptative et élimination des sommets en T

3.6.2 Élimination de la contrainte de topologie radiale

Une des lacunes principales de notre système, c'est sa dépendance à une contrainte de topologie radiale. Nous avons donc exploré quelques possibilités pour éliminer cette contrainte et ainsi rendre le système plus général.

L'approche la plus évidente est de démarrer le système avec une solution initiale fournie par l'utilisateur ou trouvée avec une autre méthode de reconstruction. Cette solution initiale permet de trouver une enveloppe extérieure ainsi qu'un coeur au modèle à reconstruire, et de bâtir le graphe de reconstruction entre ces deux enveloppes. C'est d'ailleurs l'approche choisie par Vogiatzis et al. [40] dans leur méthode de reconstruction volumique globale. En théorie, il serait possible de reconstruire des objets de n'importe quelle topologie à partir du moment où la reconstruction initiale respecte la même topologie. La nécessité d'amorcer le système avec une solution initiale relativement précise est cependant un inconvénient majeur.

Nous avons aussi songé à un système qui permettrait de déformer itérativement le volume de reconstruction selon la solution initiale. L'approche proposée consiste à utiliser la solution courante comme nouveau maillage patron, et à positionner les autres couches tout autour. Pour ce faire, on calcule la normale à chacun des sommets

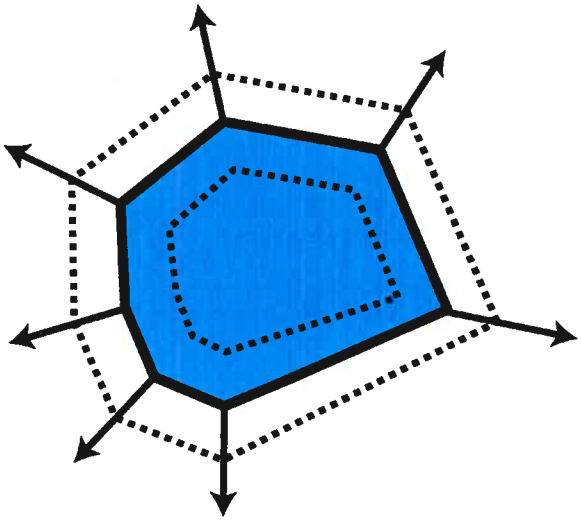


Figure 3.12: Déformation radiale du graphe

du patron, et on positionne les sommets correspondants des couches intérieures et extérieures le long de cette normale (figure 3.12). On suppose cependant que la première reconstruction s'approche de la topologie exacte de l'objet, ce qui n'est pas nécessairement le cas si cette topologie est très différente de la topologie radiale.

De plus, les deux approches proposées risquent de créer des auto-intersections dans les couches placées autour de la couche principale (figure 3.13). Or, la détection et d'élimination de ces intersections est un problème significatif en soi, qui a été exploré, entre autres, par Yoshizawa et al. [41].

De plus, si la topologie initiale est dramatiquement différente de la topologie radiale, la première itération ne sera jamais assez près de la véritable surface pour permettre la convergence du processus. Plus précisément, il y aura habituellement des pics de bruit importants là où la topologie n'est pas respectée, et ces pics viendront corrompre le processus de déformation. Des tests initiaux nous ont confirmé que le processus de déformation ne parvenait à converger que dans des cas très simples où la topologie était très proche de celle supposée. Nous avons donc jugé qu'il était

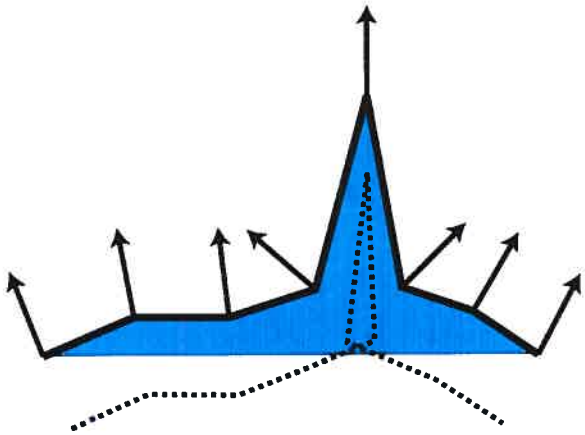


Figure 3.13: Problème de l'autointersection du graphe lors de la déformation radiale

préférable de limiter la portée de notre problème aux objets de topologie radiale, et de laisser l'assouplissement de cette contrainte à des travaux futurs.

3.7 Algorithme final

Voici finalement l'algorithme complet intégrant la méthode de base et ses raffinements. Les algorithmes 2 et 3 présentent, en pseudo-code, les deux phases de la méthode: on débute par la pré-élimination des caméras, puis on fait une série de reconstructions à basse résolution afin de raffiner la visibilité. Une fois que la visibilité est satisfaisante (le nombre d'itérations doit être choisi par l'utilisateur), on utilise la solution à basse-résolution trouvée à la phase I pour réévaluer le volume à haute-résolution et produire la reconstruction finale.

Algorithme 2: Reconstruction volumétrique: phase I à basse résolution

```
(1)  construire le volume de reconstruction à basse résolution
(2)  pour chaque voxel
(3)      éliminer les caméras du mauvais hémisphère
(4)      calculer la fonction de coût robuste
(5)  reconstruire avec l'algorithme du flot maximum
(6)  pour chaque caméra
(7)      faire un rendu du tampon de profondeur avec la solution initiale
(8)  pour  i = 0...N
(9)      pour  chaque voxel
(10)         évaluer l'occlusion et éliminer la pire caméra
(11)     calculer la fonction de coût avec les caméras restantes
(12)     reconstruire avec l'algorithme du flot maximum
(13)     pour chaque caméra
(14)         faire un rendu du tampon de profondeur avec la solution courante
```

Algorithme 3: Reconstruction volumétrique: phase II à haute résolution

```
(1)  construire le volume de reconstruction à haute résolution
(2)  pour chaque voxel
(3)      éliminer les caméras du mauvais hémisphère
(4)      pour  i = 0...N
(5)          évaluer l'occlusion avec la solution basse résolution trouvée en I
(6)          éliminer la pire caméra
(7)  calculer la fonction de coût avec les caméras restantes
(8)  reconstruire avec l'algorithme du flot maximum
(9)  afficher le résultat final
```

Chapitre 4

MÉTHODOLOGIE ET RÉSULTATS EXPÉRIMENTAUX

Nous présentons ici notre démarche expérimentale, ainsi que les résultats obtenus pour l'algorithme décrit à la section précédente. Les résultats se divisent en deux catégories: dans un premier temps, les résultats obtenus à partir d'images de synthèse, puis ceux obtenus à partir de photographies de scènes réelles.

4.1 Méthodologie pour les tests synthétiques

Pour l'approche synthétique, nous avons conçu une application permettant de générer des images calibrées de scènes synthétiques simples (figure 4.1). Cet utilitaire, basé sur l'API OpenGL, permet de charger des modèles en format Videoscape (une variante simplifiée du format OBJ), de placer des caméras autour de la scène et d'enregistrer les images vues par ces caméras ainsi que leurs paramètres de calibration.

4.1.1 Placement des caméras

Les caméras sont placées de façon quasi uniforme sur une sphère englobant le volume de reconstruction et pointent vers le centre du volume. L'algorithme de placement des caméras [29] (voir algorithme 4) est initialisé avec un nombre de base, correspondant à la moitié du nombre de caméras devant être placées à l'équateur de la sphère. À partir de ce nombre de base, on peut calculer la distance d'arc désirée entre deux caméras et placer les caméras quasi uniformément sur la sphère. Pour tous les tests, nous avons utilisé 5 comme nombre de base, pour un total de 30 caméras autour de la scène.

Algorithme 4: Placement quasi uniforme des caméras sur la sphère

Entrée: k, où 2k est le nombre de caméras à l'équateur, et r, la distance des caméras au centre du volume

Sortie: Vecteur de caméras positionnées

(1) $\Delta\Phi = \frac{\Pi}{k}$

(2) $L_{arc} = r \cdot \Delta\Phi$

(3) **pour** $\Phi = 0...\Pi$

(4) $\quad n_{camerassurcercle} = \lfloor \frac{2\Pi r}{L_{arc}} \rfloor$

(5) $\quad \Delta\Theta = \frac{2\Pi}{n_{camerassurcercle}}$

(6) $\quad$ **pour** $\Theta = 0...2\Pi$

(7) $\quad\quad x = r \cdot \cos(\Theta)\sin(\Phi)$

(8) $\quad\quad y = r \cdot \sin(\Theta)\sin(\Phi)$

(9) $\quad\quad z = r \cdot \cos(\Phi)$

(10) $\quad\quad INSERE\_CAMERA(x, y, z)$

(11) $\quad\quad \Theta+ = \Delta\Theta$

(12) $\quad \Phi+ = \Delta\Phi$

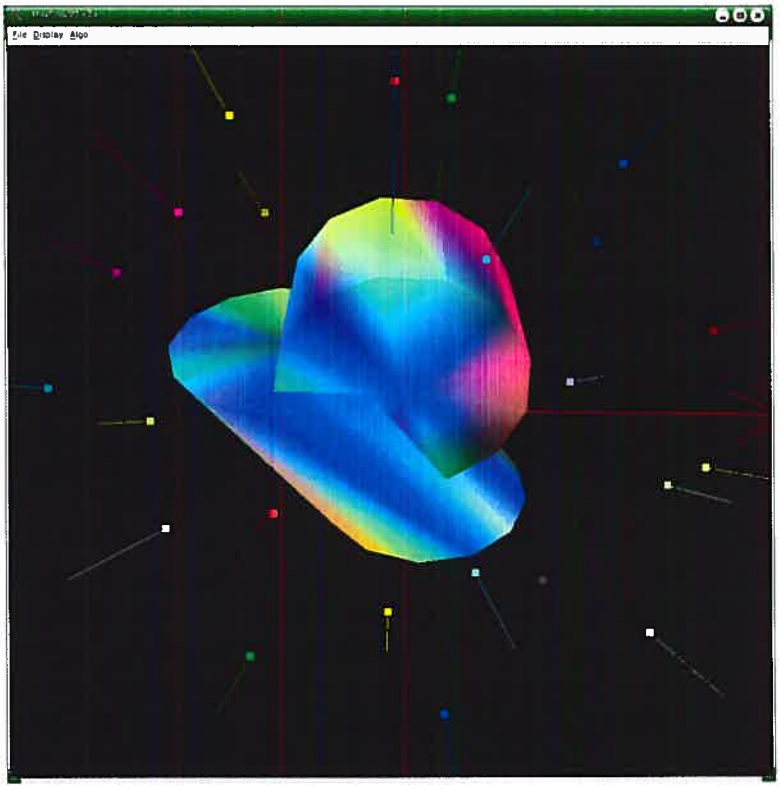


Figure 4.1: Utilitaire permettant de générer des images synthétiques. Les lignes colorées avec une "tête d'épingle" représentent la position et l'orientation des caméras.

4.1.2 Calibration des caméras

Comme nous connaissons exactement la position et l'orientation de chaque caméra de la scène, il est facile de reconstruire leurs matrices de calibration. Nous utilisons les équations suivantes [37]:

Soit R et T, les matrices de transformation correspondant à la rotation et à la translation de la caméra par rapport à sa position d'origine (centre de la caméra à $(0, 0, 0)$, axe optique pointant vers $-Z$)

f, la longueur focale

(o_x, o_y), les coordonnées en pixels du centre de l'image

(s_x, s_y), la taille effective d'un pixel

$$M_{int} = \begin{pmatrix} -\frac{f}{s_x} & 0 & o_x & 0 \\ 0 & -\frac{f}{s_y} & o_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

$$M_{ext} = T \cdot R \tag{4.2}$$

$$M = M_{int} \cdot M_{ext} \tag{4.3}$$

4.1.3 Choix des modèles

Nous proposons quatre modèles de base présentant toute une gamme de difficultés pour le processus de reconstruction (figure 4.2).

1. Poire

   - Géométrie simple, peu de détails mis à part la queue de la poire
   - Contrainte radiale parfaitement respectée
   - Pas de discontinuités ou de surfaces planes difficiles à reconstruire

2. Cube

   - Géométrie simple
   - Contrainte radiale parfaitement respectée

- Combinaison de surfaces lisses et d'arêtes bien définies

3. Canon

   - Géométrie un peu plus complexe
   - Contrainte radiale parfaitement respectée
   - Combinaison de surfaces lisses et d'arêtes bien définies

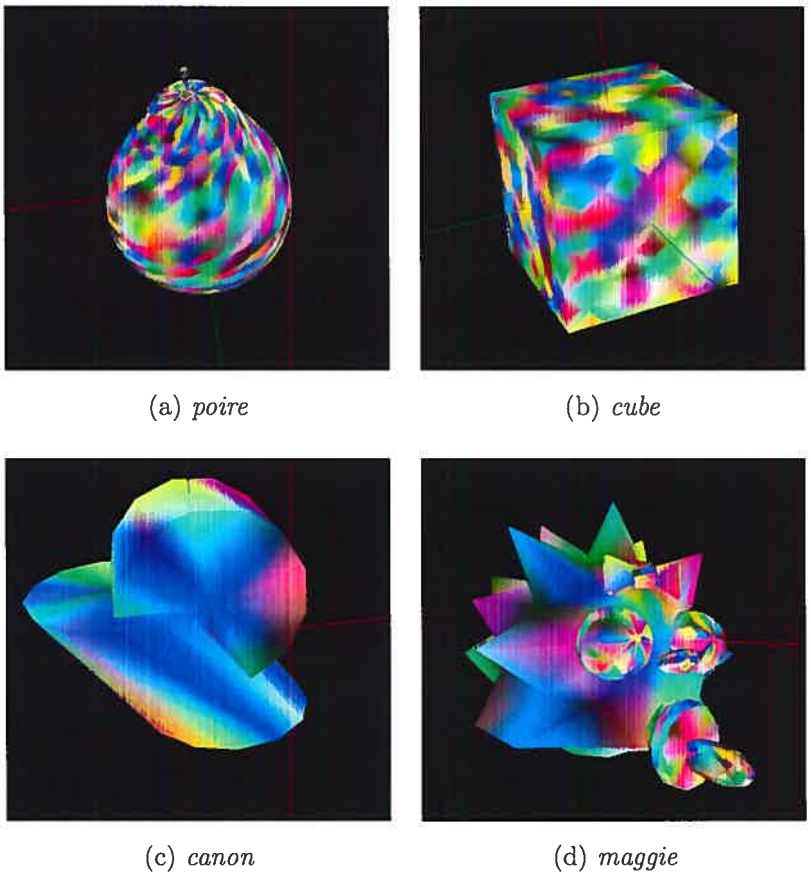
4. Maggie

   - Géométrie plus complexe en terme de nombres de surfaces avec plusieurs détails
   - Contrainte radiale généralement respectée, mais avec quelques exceptions

4.1.4 Surface des modèles

On donne aux modèles synthétiques une surface qu'on pourrait qualifier de très bien adaptée pour le processus de reconstruction. Tout d'abord, la surface est parfaitement lambertienne, ce qui concorde avec l'hypothèse posée à la section 1.2. De plus, chaque sommet du modèle est coloré avec une couleur aléatoire et les triangles sont rendus avec un mode d'interpolation de couleurs, ce qui donne une texture importante et peu répétitive à la surface. Lorsque les modèles présentent de très gros triangles qui pourraient manquer de texture, ils sont subdivisés à la main comme c'est le cas pour le modèle du cube. Toutes les reconstructions sont faites à partir de trente images avec une résolution de 400×400 pixels.

Certains tests particuliers ont été effectués avec des surfaces plus difficiles à reconstruire, et nous présenterons les résultats de ces tests aux sections 4.2.5 et 4.2.7.



(a) *poire* (b) *cube*

(c) *canon* (d) *maggie*

Figure 4.2: Modèles synthétiques utilisés

4.1.5 Mesure de l'occlusion

L'occlusion est mesurée en faisant un rendu OpenGL du tampon de profondeur. On convertit la valeur du tampon en distance dans le système de coordonnées de la caméra par l'équation suivante:

$$z_{solution} = \frac{z_{near} \cdot z_{far}}{z_{far} - d \cdot (z_{far} - z_{near})} \tag{4.4}$$

où z_{far} et z_{near} sont les profondeurs des plans de découpage (*clipping planes*) avant et arrière

et d est la valeur du tampon de profondeur.

4.2 Résultats synthétiques

Nous présentons dans cette section les résultats obtenus avec les images synthétiques dans des conditions très favorables, puis dans divers cas limites: variations du lissage et de la résolution, performances de l'algorithme avec et sans la technique multi-résolutions, présence de bruit et d'erreurs de calibration, présence de spécularités... Notons que la luminosité et le contraste des images de résultats ont été ajustés pour permettre une bonne impression.

4.2.1 Résultats de base de l'algorithme de reconstruction

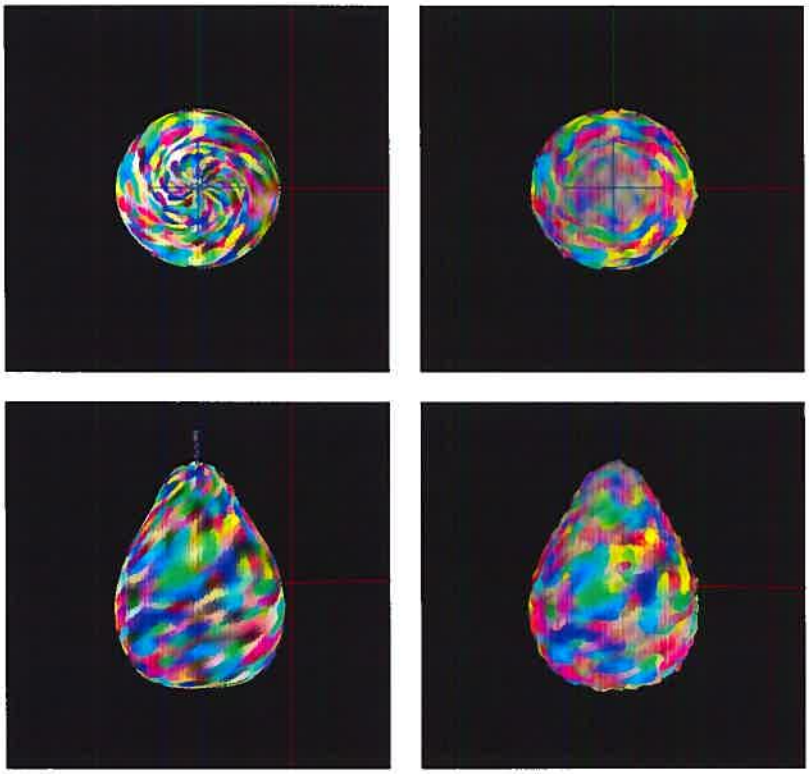
Poire

Paramètres des tests:

- 128 100 voxels, 2 562 noeuds par couche $\times$ 50 couches
- 10 itérations
- Facteur de lissage $= 0,008$
- Fonction de coût robuste

De manière prévisible étant donné sa simplicité, la reconstruction de ce modèle (figure 4.3) se fait assez facilement. De plus, comme il y a peu de détails, il est possible de choisir un lissage plus élevé que pour les modèles qui suivent.

On peut constater que la tige de la poire n'est pas reconstruite; il s'agit probablement d'un détail trop fin pour être distingué d'une erreur de mise en correspondance, et il est éliminé par le processus de lissage. Mis à part cette erreur, la forme générale est bien reconstruite et il n'y a pas de pics de bruit majeurs. Notons que la couleur attribuée à un sommet est une moyenne des couleurs des pixels sur lesquels le voxel correspondant se projette. Ceci explique pourquoi les couleurs du modèle à ses extrémités sont imprécises puisque les voxels à cet endroit couvrent chacun plusieurs triangles de couleurs différentes.



original reconstruction

Figure 4.3: Reconstruction du modèle *poire*

Cube

Paramètres des tests:

- 128 100 voxels, 2 562 noeuds par couche × 50 couches
- 10 itérations
- Facteur de lissage = 0,005
- Fonction de coût robuste

Cette reconstruction (figure 4.4) illustre bien une des limitations majeures de l'approche du flot maximum: afin de préserver les arêtes du cube, on doit tolérer une certaine quantité de bruit. De plus, comme nos voxels sont discrets et ne sont de surcroît pas orientés selon les faces du cube, il sera impossible d'obtenir une surface parfaitement plane. Il est important de bien distinguer ce bruit – qu'on appellera *bruit de discrétisation* – des erreurs dues à une reconstruction erronée. Dans le premier cas, on pourrait corriger ce bruit à l'affichage, en moyennant les normales par exemple, ce qui le rend beaucoup moins indésirable.

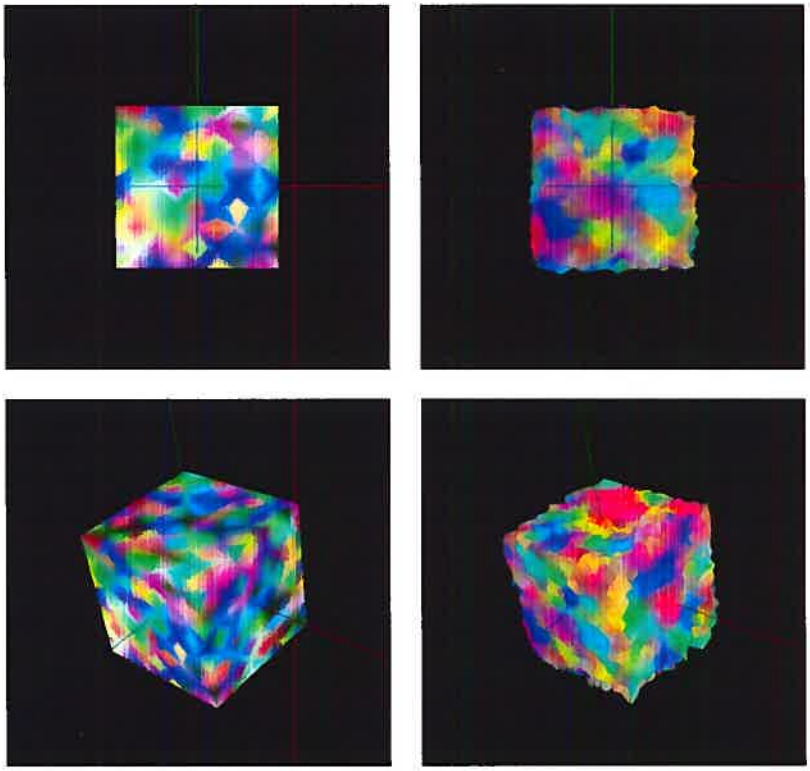
Étant donné ces difficultés, le système reconstruit relativement bien l'objet. La taille du cube n'est ni surestimée ni sous-estimée et la forme est bien respectée, mis à part un bruit de faible intensité et de haute fréquence sur la surface et les arêtes.

Canon

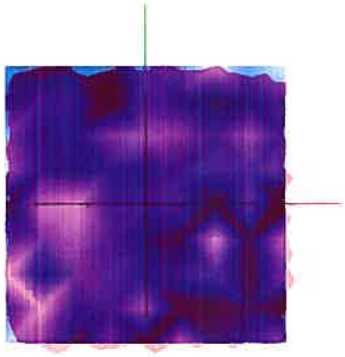
Paramètres des tests:

- 409 680 voxels, 10 242 noeuds par couche × 40 couches
- 9 itérations
- Facteur de lissage = 0,005
- Fonction de coût robuste

On observe ici (figure 4.5) que le modèle est bien reconstruit presque partout, mais qu'il présente une crevasse importante sur une de ses arêtes. Cette crevasse est



original reconstruction



surimposition de l'original et de la reconstruction

Figure 4.4: Reconstruction du modèle *cube*

vraisemblablement due à une ambiguité de la fonction de coût. Pour expliquer ce phénomène, nous spéculons que la fonction présente, par hasard, un coût très faible à une couche intérieure du graphe, et que ce minimum local est trop fort pour être éliminé par le lissage. Un lissage supérieur gommerait cependant les arêtes du modèle.

Maggie

Paramètres des tests:

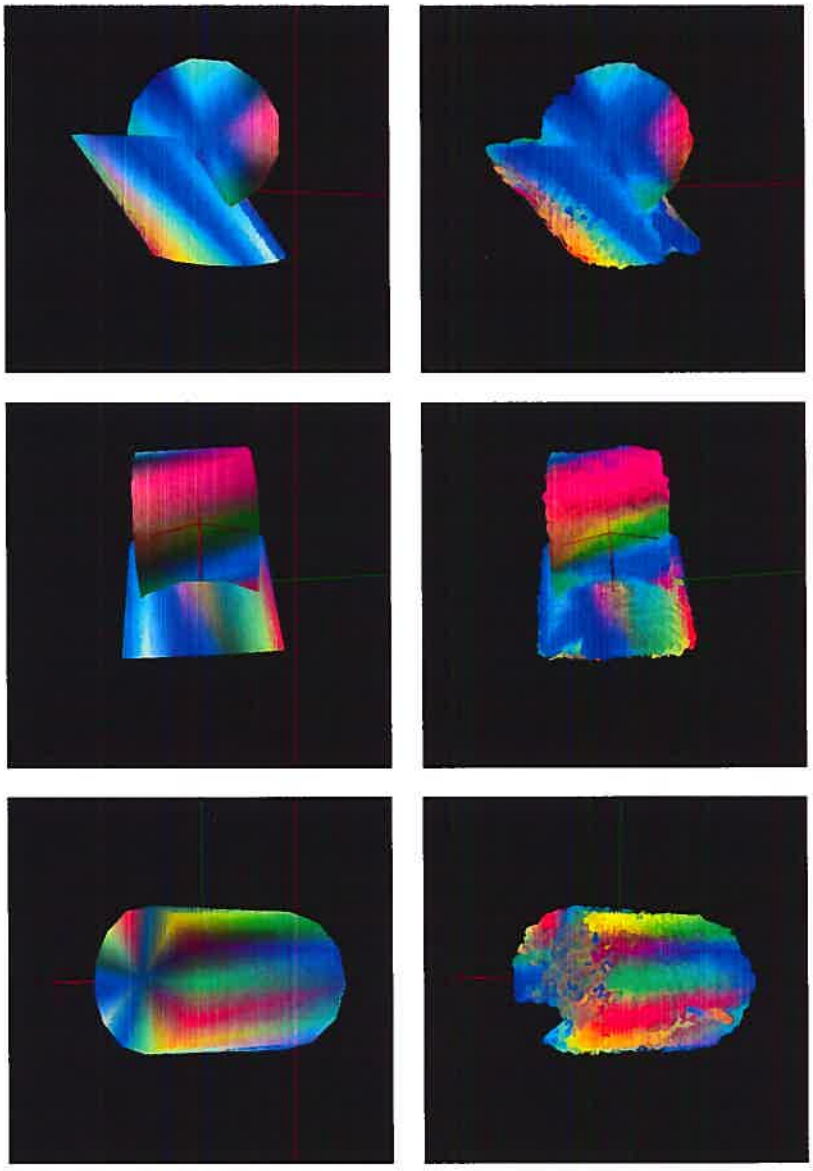
- 409 680 voxels, 10 242 noeuds par couche $\times$ 40 couches
- 10 itérations
- Facteur de lissage $= 0,005$
- Fonction de coût robuste

Sur notre modèle le plus complexe (figure 4.6), on commence à sentir les limitations de notre approche. On voit très bien sur la seconde image que la sucette du bébé, qui ne respecte pas la contrainte de topologie radiale, est tout simplement éliminée. De plus, il est relativement difficile d'obtenir une surface non bruitée tout en préservant les détails de la surface.

Nous avons fait le compromis d'utiliser un lissage moyen (le même que celui utilisé pour les modèles *canon* et *cube*) permettant de bien reconstruire la plupart des détails (boucle sur le front, pointes des cheveux), mais lissant les détails les plus fins comme le nez. Comme dans le modèle du cube, le bruit de discrétisation devient ici un problème non négligeable.

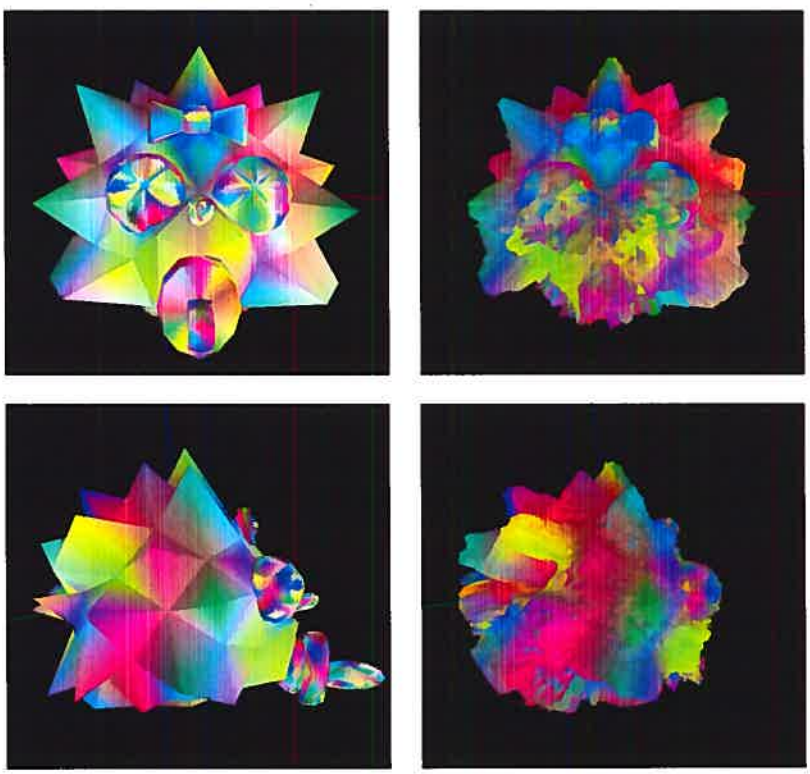
4.2.2 Taille du volume de reconstruction et temps de calcul

On présente ici la reconstruction d'un même modèle à différentes résolutions pour illustrer les performances de l'algorithme en très basse, basse, moyenne et haute qualité (figure 4.8), ainsi que les temps de calcul (tableau 4.1) sur une machine Pentium M 1,6 GHz avec 1 Go de mémoire vive. De plus, le facteur de lissage utilisé est le même



original reconstruction

Figure 4.5: Reconstruction du modèle *canon*



original reconstruction

Figure 4.6: Reconstruction du modèle *maggie*

résolution	nb voxels/couche	nb couches	nb voxels	temps de calcul (sec)
très basse	162	30	4 860	17
basse	642	40	25 680	67
moyenne	2 562	50	128 100	379
haute	10 242	40	614 520	2 204

Table 4.1: Temps de calcul de l'algorithme à différentes résolutions

dans toutes les reconstructions. Il est important de noter que le système n'a pas été optimisé en fonction de ses performances en temps. Il ne s'agit donc que d'une mesure générale pour donner une idée de l'ordre de l'algorithme. Les temps de calcul ne tiennent pas non plus compte du temps nécessaire à l'initialisation de l'application et la construction du graphe puisque cette étape est indépendante de l'algorithme lui-même. De plus, l'initialisation n'a pas été optimisée, ce qui fausserait grandement les résultats.

On constate, dans un premier temps, que les résultats sont acceptables même à très basse résolution. On perd évidemment tous les détails de la surface, mais le système converge quand même vers une forme générale semblable à celle du sujet. On peut donc adapter la reconstruction aux besoins de l'application et n'obtenir qu'une forme très grossière si, par exemple, l'objet sera vu de très loin ou encore qu'on ne veut qu'estimer de manière générale son centre de gravité. On n'a pas non plus à ajuster le facteur de lissage en fonction de la résolution du système.

L'autre constatation importante, c'est que les temps de calcul montrent empiriquement que l'algorithme est linéaire en fonction du nombre de voxels dans un cas typique. Les temps de calcul à d'autres résolutions (figure 4.7) confirment cette tendance. Nous avons constaté que l'essentiel du temps de calcul était consacré à la projection des voxels, et que l'opération de projection elle-même se fait en temps constant. Il est donc logique que l'algorithme soit du même ordre.

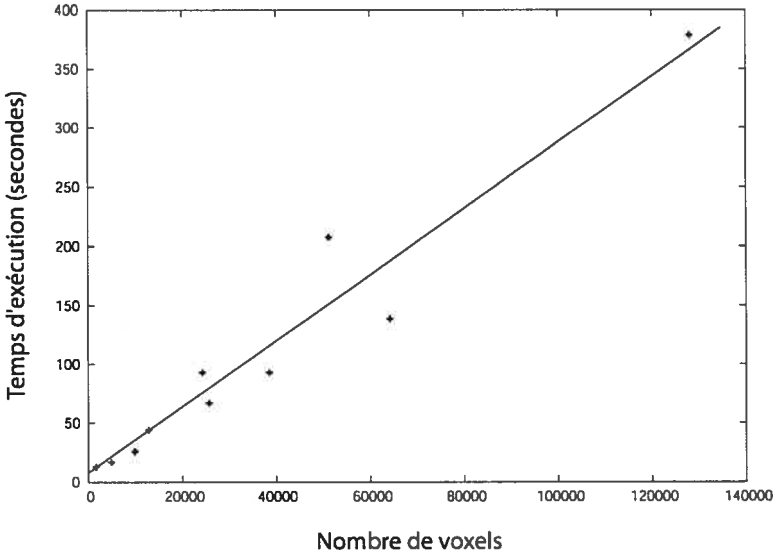


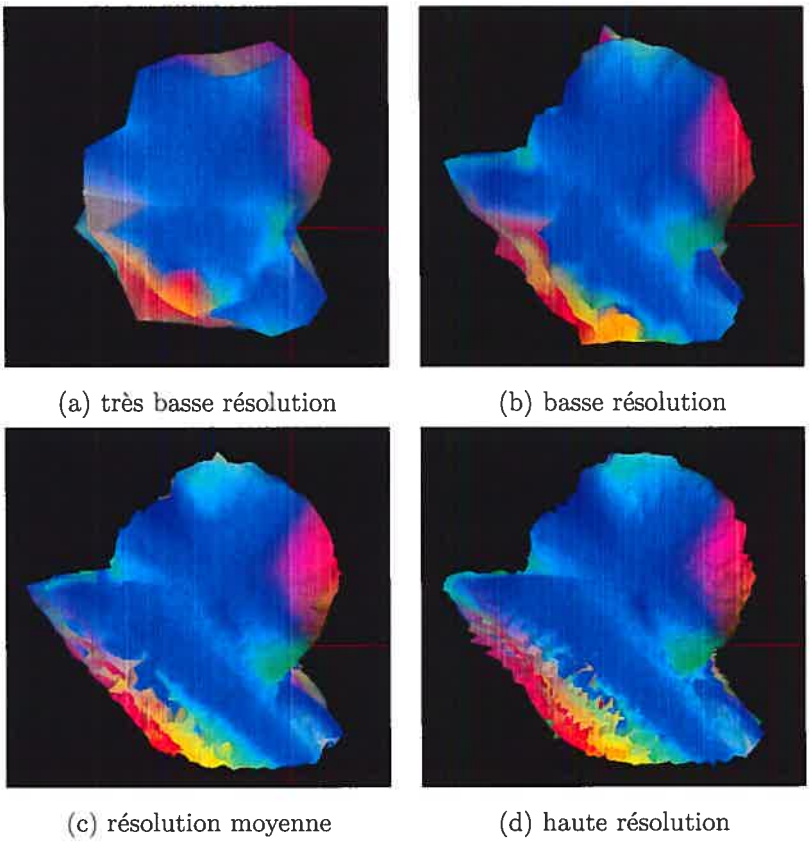
Figure 4.7: Estimation de l'ordre de l'algorithme

4.2.3 Influence du lissage

La figure 4.9 illustre les résultats obtenus lorsqu'on reconstruit une même scène avec différents paramètres de lissage. On peut constater, dans un premier temps, qu'un lissage qui tend vers l'infini donne une reconstruction parfaitement sphérique alors qu'un lissage nul donne une reconstruction très bruitée. D'autre part, on peut observer que le système est assez résistant aux variations du lissage. Il n'est donc pas nécessaire d'utiliser une valeur de lissage absolument parfaite pour obtenir une reconstruction valide. Notons que de façon générale, un même lissage donne de bons résultats pour la plupart des scènes, et ne doit être ajusté que lorsqu'on sait que la scène est très simple et qu'on veut éliminer tout bruit, ou qu'au contraire on cherche à reconstruire de petits détails.

Paramètres des tests:

- 128 100 voxels, 2 562 noeuds par couche $\times$ 50 couches
- 10 itérations



(a) très basse résolution (b) basse résolution

(c) résolution moyenne (d) haute résolution

Figure 4.8: Reconstructions à différentes résolutions

- Facteur de lissage variable
- Fonction de coût robuste

4.2.4 Processus multi-résolutions

Nous testons ici l'approche à résolutions multiples décrite à la section 3.5.2. Dans un premier cas, on calcule 5 itérations de l'algorithme de base en haute résolution. Dans le second, on calcule 5 itérations en basse résolution, puis on recalcule le flot maximum avec un graphe haute résolution en utilisant le modèle basse résolution pour évaluer la visibilité.

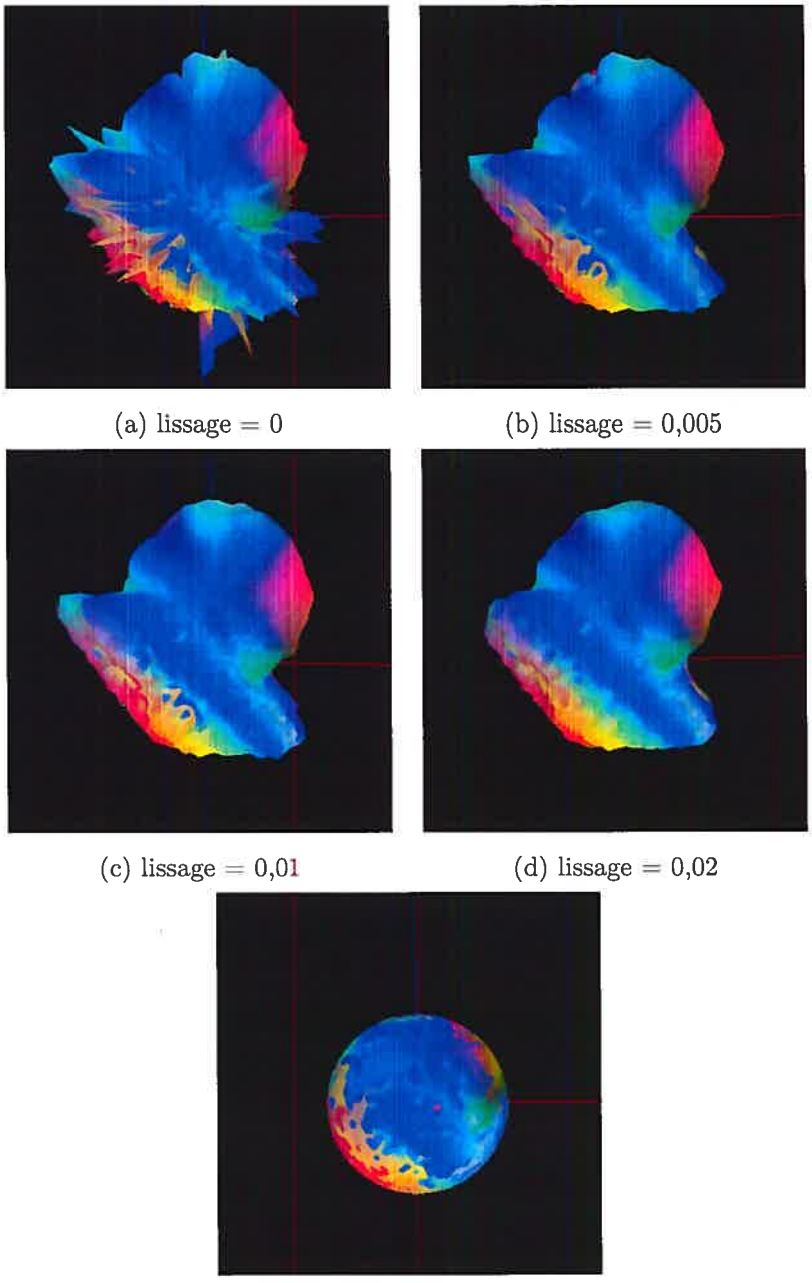
Les résultats obtenus sont illustrés à la figure 4.10. On constate que l'approche multi-résolutions donne des résultats très similaires à ceux de l'approche normale en seulement 76 % du temps. Notons qu'ici, les temps de calcul tiennent compte du temps d'initialisation du graphe puisque dans le cas de la reconstruction multi-caméra, cette opération doit être faite deux fois avec des graphes différents. Il est donc important d'en tenir compte. Ceci explique aussi pourquoi les temps de calcul diffèrent beaucoup des résultats de la section précédente.

4.2.5 Performances en présence de spécularités

On simule la présence de reflets sur la surface en ajoutant une composante spéculaire aux propriétés du matériel en OpenGL. On peut ainsi tester les performances du système lorsque l'hypothèse de surface lambertienne n'est pas respectée. La figure 4.11 permet de comparer la scène avec spécularités par rapport à la scène lambertienne généralement utilisée. Les résultats obtenus sont illustrés à la figure 4.12. On constate que la surface présente quelques erreurs là où les reflets sont présents, mais que globablement la forme demeure bien reconstruite.

Paramètres des tests:

- 128 100 voxels, 2 562 noeuds par couche $\times$ 50 couches



(a) lissage = 0

(b) lissage = 0,005

(c) lissage = 0,01

(d) lissage = 0,02

(e) lissage = ∞ (la valeur de 1 a été utilisée)

Figure 4.9: Influence du facteur de lissage

- 5 itérations
- Facteur de lissage $= 0,005$
- Fonction de coût robuste

4.2.6 Performances en présence d'erreurs de calibration

On simule la présence d'erreurs de calibration en ajoutant du bruit aux composantes de chacune des caméras. Plus précisément, on perturbe les propriétés de la caméra comme suit:

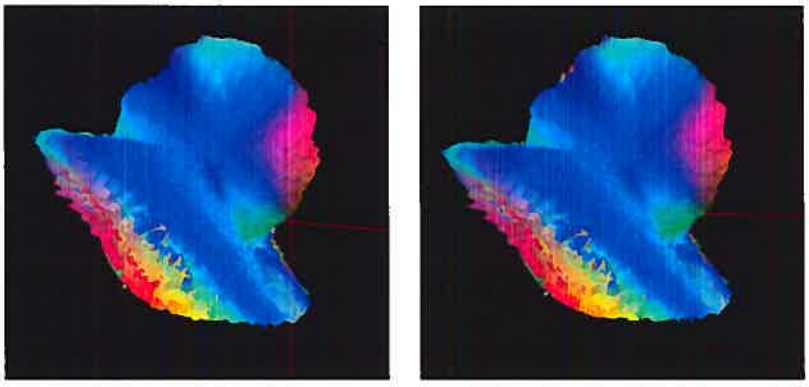
- On ajoute une rotation sur chacun des axes z, y, x avec un angle aléatoire donné selon une distribution gaussienne avec $\mu = 0$ et $\sigma^2 = 0,01$
- On ajoute une translation (dx, dy, dz) où chacune des composantes est donnée par la même distribution gaussienne que pour la rotation
- On perturbe la longueur focale et l'angle de vue avec un bruit gaussien décrit par $\mu = 0$ et $\sigma^2 = 1$ %

La matrice de projection résultante appliquée sur un volume de reconstruction typique (sphère de trois unités de rayon, image 200×200) donne une erreur moyenne de 1,3 pixel et une erreur maximale de 4 pixels, ce qui est environ quatre fois plus élevé que le bruit de calibration de nos images réelles (entre 1 et 3 pixels pour une image 768×484 avec la séquence *cactus* présentée à la section 4.4).

On obtient les résultats illustrés à la figure 4.13. On constate que même avec de larges erreurs de calibration, la surface de l'objet demeure relativement bien reconstruite. De plus, les pointes de bruit présentes pourraient facilement être gommées avec un lissage plus important (au détriment, bien entendu, des détails de la surface).

Paramètres des tests:

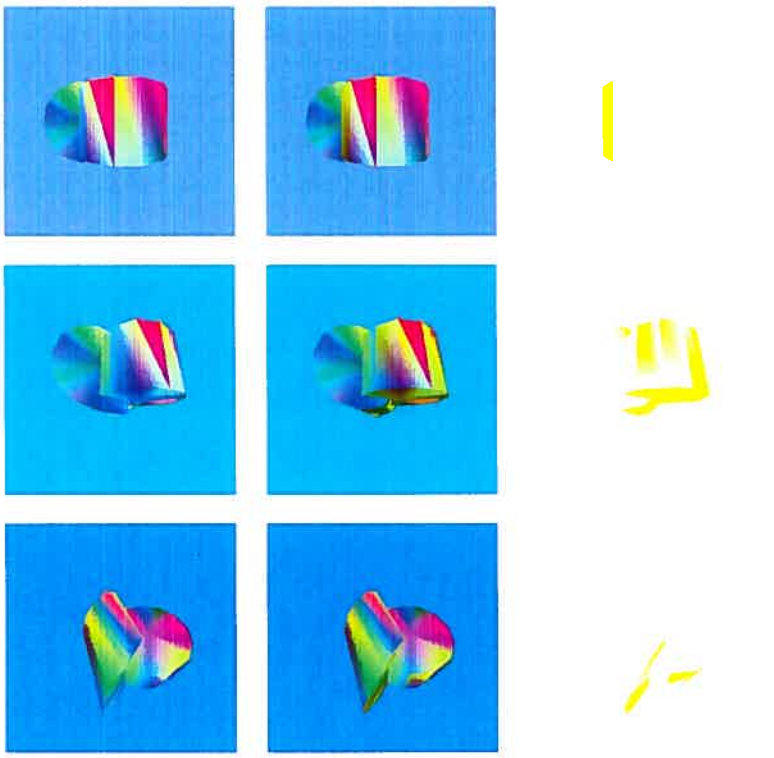
- 128 100 voxels, 2 562 noeuds par couche $\times$ 50 couches
- 5 itérations



(a) sans le processus multi-résolutions, 1h41 (6094 secondes)

(b) avec le processus multi-résolutions, 1h17 (4630 secondes)

Figure 4.10: Performance du système multi-résolutions

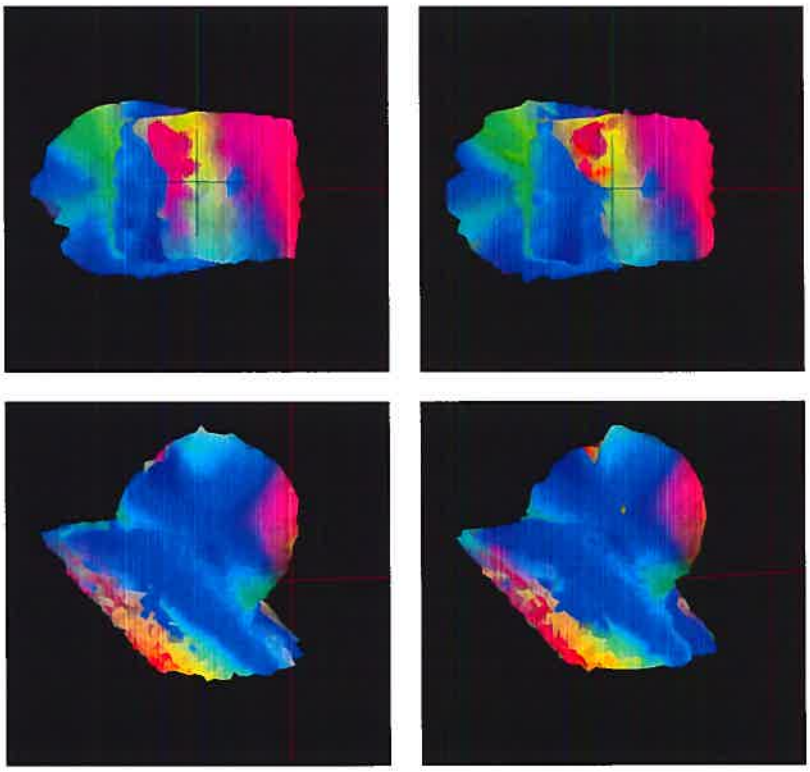


sans spécularités

avec spécularités

différence entre les deux images

Figure 4.11: Trois des trente images de la scène *canon*, sans et avec spécularités



sans spécularités avec spécularités

Figure 4.12: Influence des spécularités sur la reconstruction

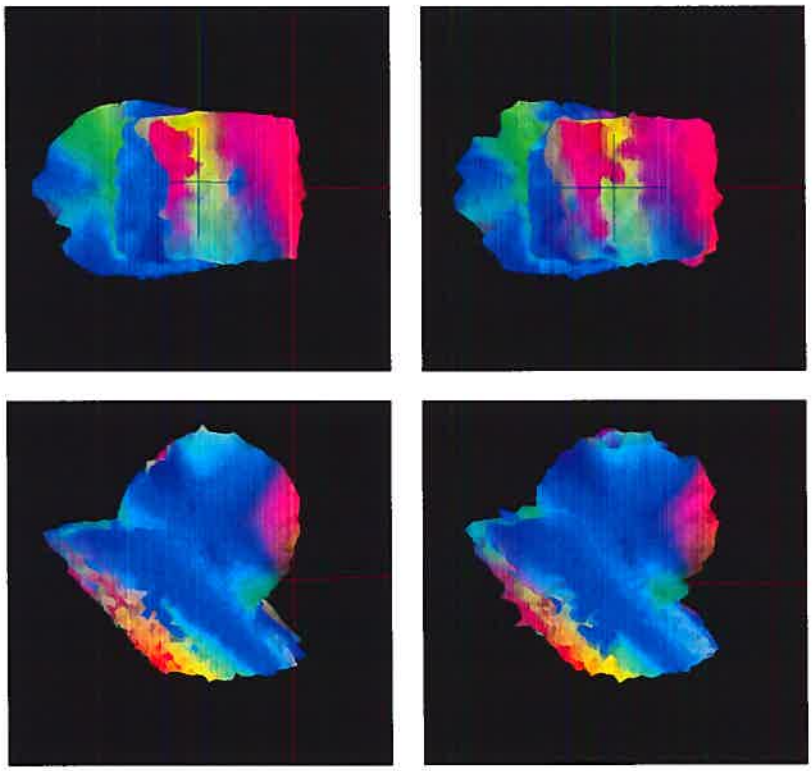
- Facteur de lissage $= 0,005$
- Fonction de coût robuste

4.2.7 Performances en présence de bruit dans les images

On simule la présence de bruit en ajoutant un bruit gaussien de moyenne 10 et d'écart-type 9 (sur une échelle de 0 à 255) dans chacun des canaux rouge, bleu et vert des images de synthèse. La figure 4.14 illustre le type d'images résultantes et la figure 4.15 montre les résultats obtenus. Comme pour les erreurs de calibration, on constate que le système est peu sensible au bruit dans les images.

Paramètres des tests:

- 128 100 voxels, 2 562 noeuds par couche $\times$ 50 couches
- 5 itérations
- Facteur de lissage $= 0,005$
- Fonction de coût robuste



calibration parfaite calibration bruitée

Figure 4.13: Influence des erreurs de calibration sur la reconstruction

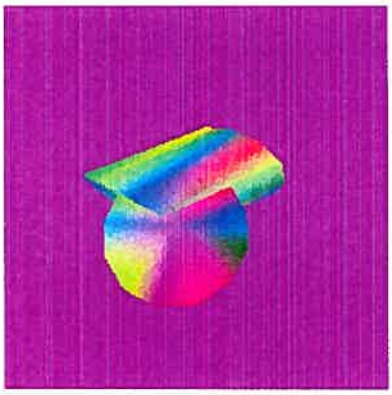
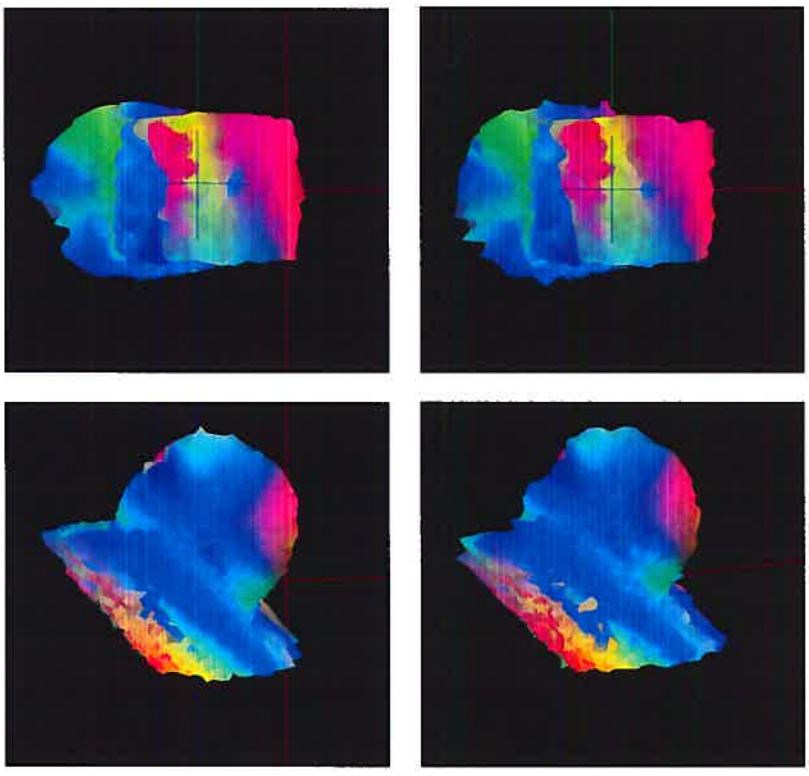


Figure 4.14: Exemple d'image utilisée pour simuler la présence de bruit



sans bruit images bruitées

Figure 4.15: Influence du bruit dans les images sur la reconstruction

4.3 *Méthodologie pour les tests avec des images réelles*

4.3.1 Images utilisées

La banque d'images utilisée a été fournie par Kutulakos [1]. Ces images sont calibrées de manière imparfaite avec l'algorithme de calibration de Tsaï [38], ce qui permet de tester le système dans des conditions réelles d'utilisation.

Deux séries d'images différentes ont été utilisées. La séquence *cactus* est composée de trente images, prises à partir d'un bras robotisé, illustrant quatre cactus dans un pot (figure 4.16). Les caméras sont placées en cercle au-dessus de la scène; on ne pourra donc pas reconstruire le dessous du pot, ou certains détails du milieu de la scène – la terre située entre les cactus notamment – qui ne sont visibles d'aucune des caméras. En revanche, la scène présente beaucoup de textures et de couleurs différentes ce qui devrait faciliter la reconstruction, et la scène respecte assez bien la contrainte de topologie radiale. En revanche, la calibration est imparfaite: on peut s'attendre à des erreurs de reprojection entre 0,5 et 1,5 pixel.

La séquence *gargoyle* est composée de seize images d'une petite sculpture prises avec une table tournante (figure 4.17). Il s'agit donc d'une séquence plus limitée, tant pour le nombre d'images que pour la variété des points de vue, mais aussi plus typique de ce qu'un utilisateur moyen peut obtenir avec une caméra unique montée sur un trépied. La scène présente moins de couleurs que la scène précédente, mais la surface de la sculpture est quand même bien texturée. Les erreurs de reprojection sont d'environ 0,5 pixel. De plus, comme la topologie radiale n'est pas respectée, nous nous concentrerons uniquement sur la reconstruction de la tête de la gargouille.

Comme l'arrière-plan était très similaire dans toutes les images, ce qui risquait de créer de fausses mises en correspondance, les images ont été grossièrement segmentées à la main et une couleur aléatoire a été appliquée pour couvrir le fond.

[1] Université de Toronto

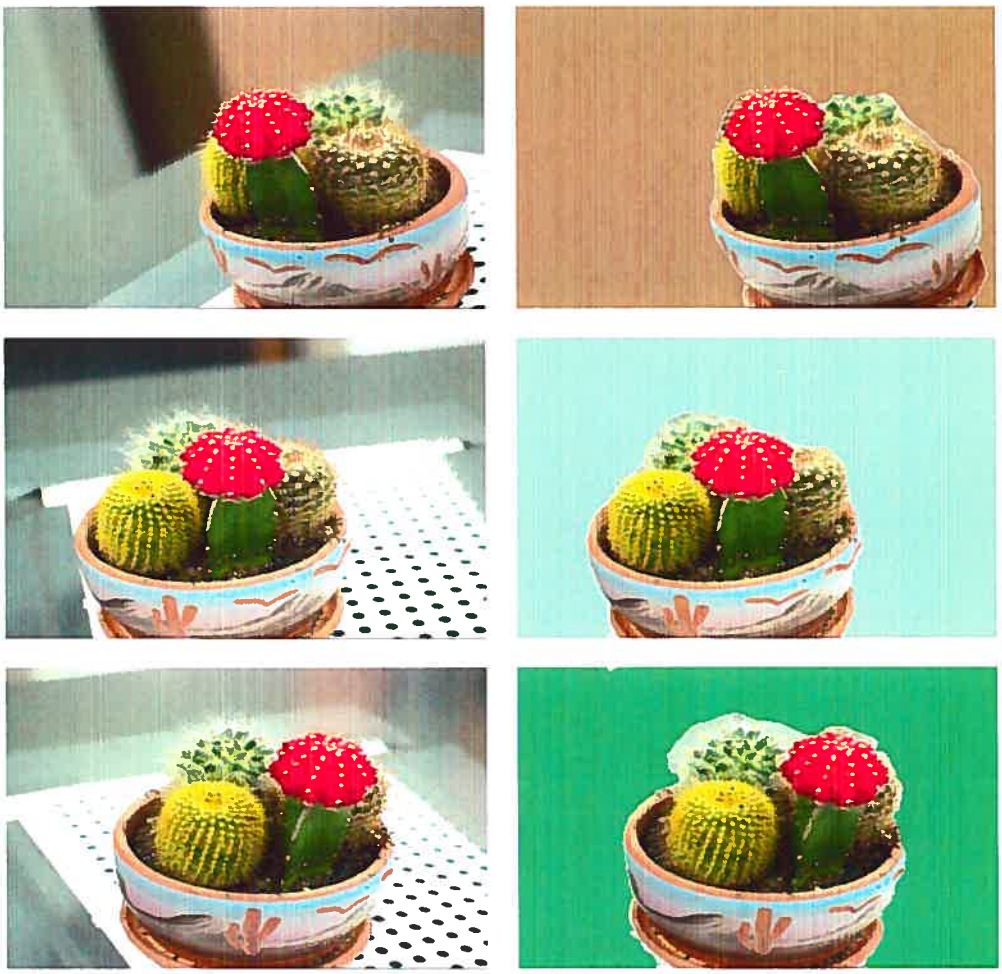


Figure 4.16: Trois des trente images de la scène *cactus*, sans et avec segmentation manuelle

Figure 4.17: Trois des seize images de la scène *gargoyle*, sans et avec segmentation manuelle

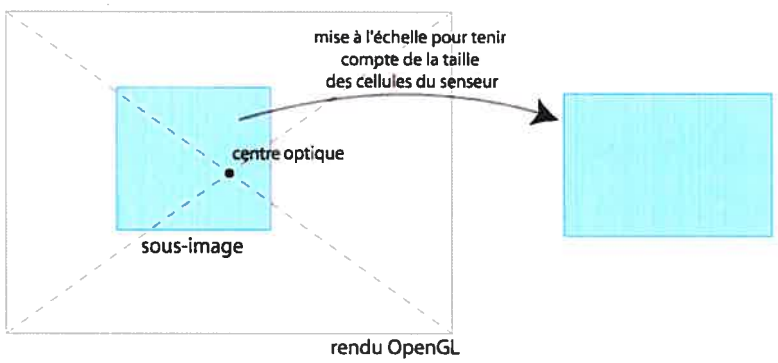


Figure 4.18: Processus de découpage et mise à l'échelle du rendu OpenGL pour mettre en correspondance l'image synthétique et l'image réelle

4.3.2 Mesure de l'occlusion

Pour mesurer l'occlusion, on reproduit la scène réelle en OpenGL et on associe une caméra synthétique à chaque image calibrée. Comme le centre optique de la caméra ne coïncide pas directement avec le centre de l'image, il est difficile d'avoir une caméra OpenGL identique à la caméra réelle. On a donc choisi de faire un rendu plus large de la scène, et découper seulement la partie de l'image correspondant à la vue de la caméra. Il faut aussi tenir compte du fait que les cellules du senseur de la caméra ne sont pas parfaitement carrées contrairement aux pixels de la caméra OpenGL; on a donc appliqué une mise à l'échelle pour trouver les pixels correspondants d'une image à l'autre (figure 4.18). On doit aussi appliquer une translation et un facteur d'échelle au système de coordonnées du monde pour rendre celui-ci compatible avec notre système de visualisation synthétique. Il s'agit là de la stratégie la plus simple pour recentrer le volume de reconstruction sur l'origine et pour le redimensionner pour notre application.

Après la première reconstruction, on peut afficher la solution courante et utiliser ce modèle pour estimer l'occlusion avec le tampon de profondeur comme c'est fait pour les modèles synthétiques.

4.4 Résultats avec des images réelles

4.4.1 Cactus

Les figures 4.19 et 4.20 présentent les résultats obtenus avec la séquence *cactus* à partir de caméras connues et de nouvelles caméras. Deux problèmes majeurs peuvent être soulignés: la reconstruction de la tête du cactus rouge et celle d'un des côtés du pot. Dans le premier cas, l'erreur est attribuable au non-respect de la contrainte radiale à cet endroit.

Le trou présent sur un des côtés du pot, près du cactus rouge, est quant à lui dû à la présence de reflets très prononcés sur la surface de l'objet. On a vu à la section 4.2.5 que le système peut tolérer une certaine quantité de reflets spéculaires. Les reflets présents ici sont cependant trop prononcés et couvrent une trop grande surface de l'objet pour que l'algorithme puisse récupérer la surface réelle de l'objet.

Ces résultats permettent aussi d'observer que l'algorithme parvient à combler les zones situées entre les cactus et sous le pot, même si ces zones ne sont vues par aucune des caméras.

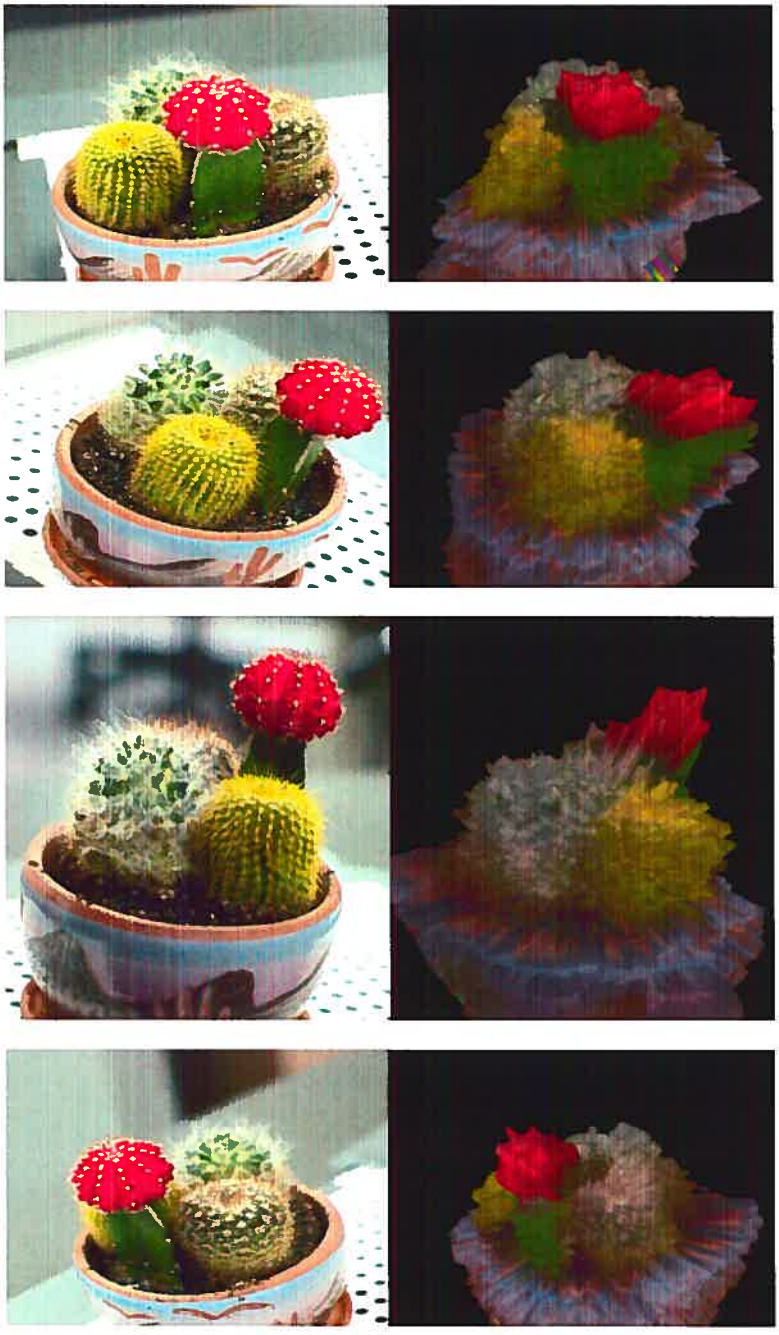


Figure 4.19: Quatre vues reconstituées de la scène *cactus* avec l'image réelle correspondante

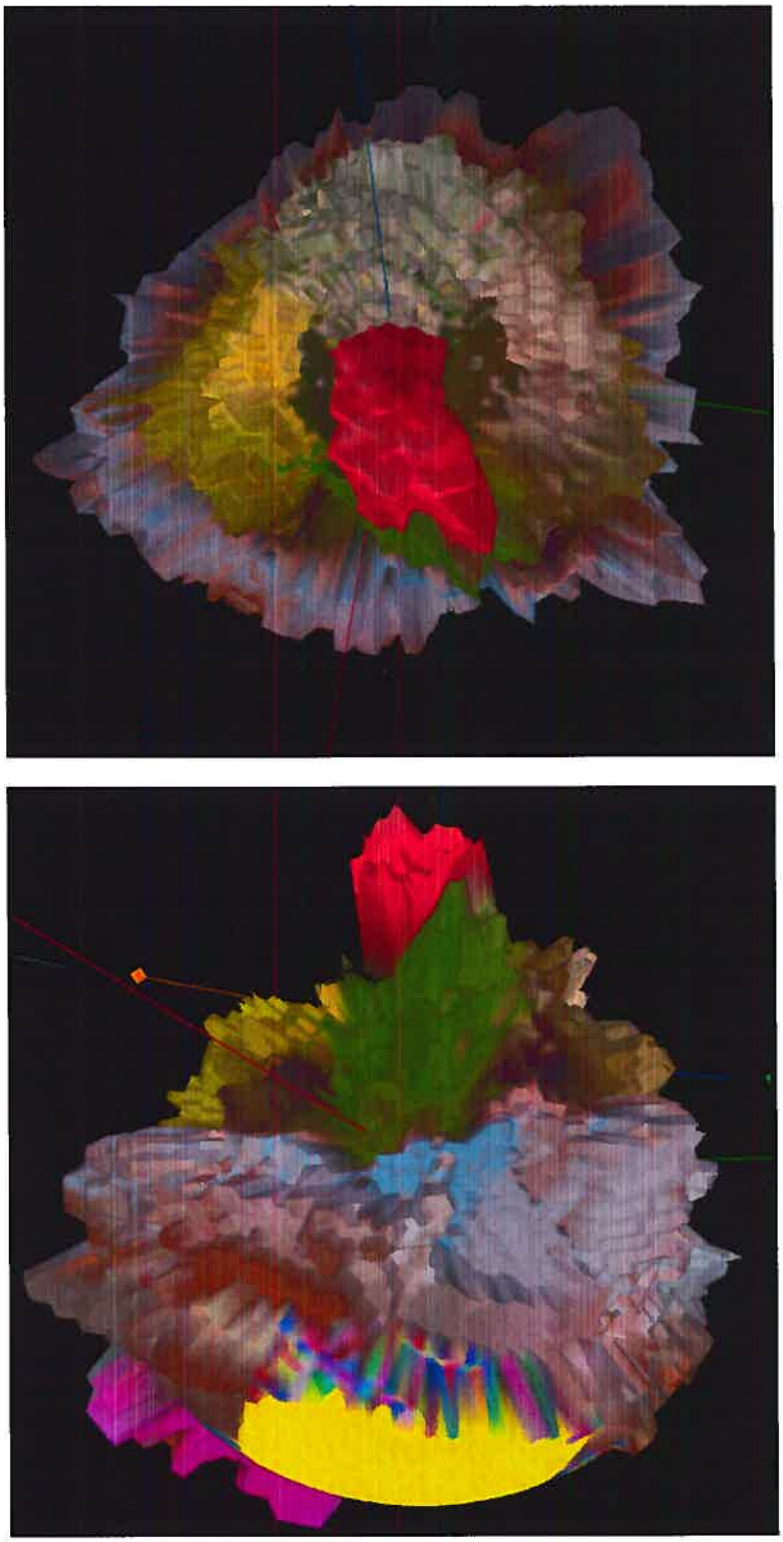


Figure 4.20: Deux nouvelles vues de la scène *cactus*

4.4.2 Gargoyle

Les résultats de la séquence *gargoyle*, aux figures 4.21 et 4.22, nous permettent d'observer également les limitations de l'algorithme quant à la contrainte de topologie radiale. On perd en effet la totalité de la main et du bras de la gargouille. En revanche, même de petits détails relativement subtils de la surface comme ceux des sourcils ou des oreilles sont bien reconstruits en autant qu'ils respectent la contrainte topologique.

On observe aussi un pic de bruit sur le dessus de la tête de la gargouille, probablement explicable par le fait que cette zone est très peu couverte par les caméras. Mis à part ces quelques erreurs et le manque de détails inhérents à la résolution, le modèle est assez fidèle, et ce même pour des vues complètement nouvelles (figure 4.22). On peut donc conclure que notre méthode est efficace même avec des points de vue limités.

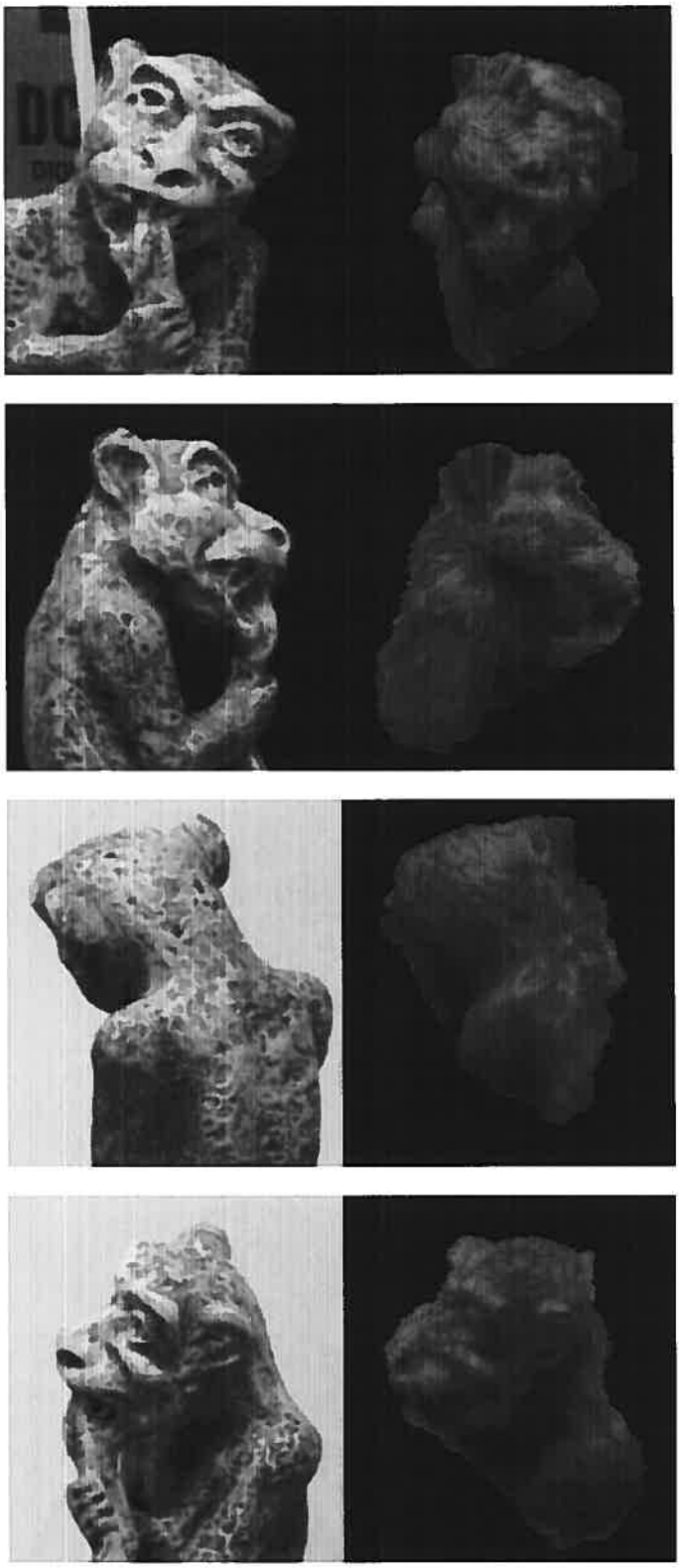


Figure 4.21: Quatre vues reconstituées de la scène *gargoyle* avec l'image réelle correspondante

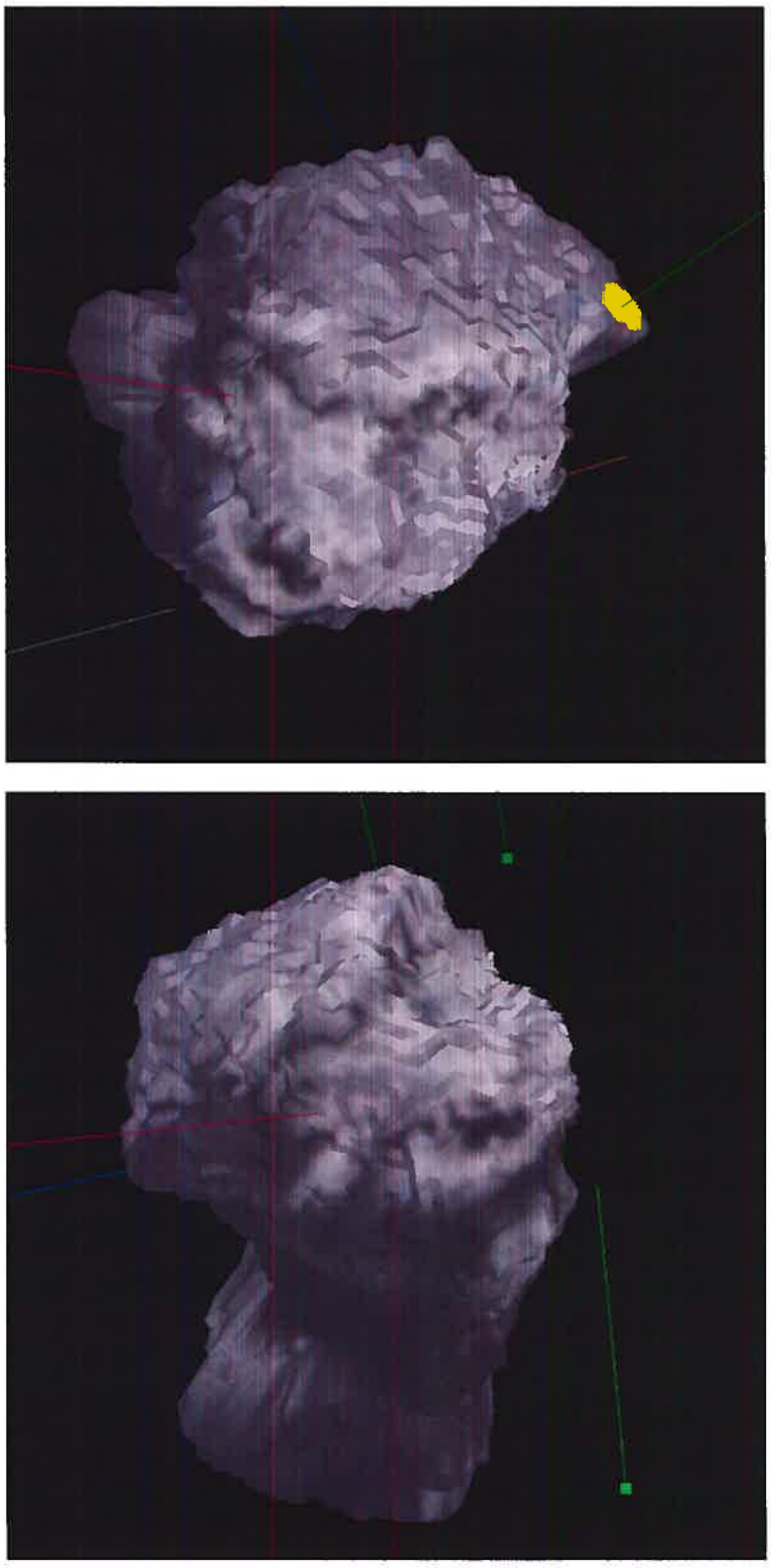


Figure 4.22: Deux nouvelles vues de la scène *gargoyle*

Chapitre 5

CONCLUSION

5.1 Discussion des résultats

Les résultats synthétiques nous permettent de constater que notre méthode donne des résultats assez fidèles lorsque les containtes initiales, particulièrement la contrainte radiale, sont bien respectées, que l'objet présente une texture importante et que la résolution du volume de reconstruction est suffisante. De plus, on observe que la reconstruction est peu affectée par le bruit de calibration ou les spécularités. On constate aussi que le système est peu sensible au facteur de lissage puisqu'un même facteur a été utilisé pour la grande majorité des reconstructions indépendamment de la résolution ou du niveau de détail. En ce sens, notre méthode se distingue des approches basées sur le *space carving* qui sont extrêmement sensibles au seuil choisi pour la fonction de coût.

On observe aussi, dans les résultats basés sur des images réelles, que la technique du flot maximum dans un graphe permet de combler les sections du modèle qui ne sont vus par aucune caméra grâce à son mécanisme de lissage. Même là où des sections du modèle ne respectent pas la contrainte radiale ou sont invisibles sur les images, la reconstruction reste cohérente et lisse. Nos résultats réels présentent toutefois encore beaucoup de bruit et les modèles présentent des trous importants lorsque la fonction de coût est peu discriminante sur toute une section du modèle. De plus, les besoins en mémoire de l'algorithme nous empêchent d'obtenir une résolution vraiment suffisante.

5.2 Contributions principales de la recherche

Notre algorithme apporte trois contributions majeures aux techniques actuelles de reconstruction 3D. Dans un premier temps, nous proposons un volume de reconstruction unique qui remplit en quelque sorte le vide laissé entre les approches volumétriques classiques, basées sur un volume cubique régulier, et les approches basées sur l'évolution radiale d'une surface comme les *level sets* [14] ou les travaux de Vogiatzis et al. [40].

Notre mécanisme itératif de gestion des occlusions constitue une autre innovation majeure. Les mécanismes classiques de reconstruction 3D appliquent une stratégie conservatrice à l'évaluation de la visibilité et n'utilisent que les caméras qui sont assurées de voir un voxel dans le calcul de sa fonction de coût. Nous proposons plutôt une approche inverse: notre système utilise au départ toutes les caméras de la scène et élimine celles-ci au fur et à mesure que la solution se précise. Cet approche a l'avantage d'éliminer l'aspect séquentiel des approches classiques; on n'est pas obligé de déterminer l'état des voxels dans un ordre prédéterminé.

Sans ces deux innovations, il nous aurait été impossible de mettre en oeuvre notre objectif principal: l'extension de la méthode du flot maximum dans un graphe à la reconstruction volumétrique. Le volume de reconstruction proposé permet au flot de circuler de l'extérieur vers l'intérieur de la scène et de saturer là où se trouve la surface. La gestion itérative des occlusions, quant à elle, permet d'évaluer la fonction de coût sur tout le graphe et de faire une recherche globale plutôt que locale de la solution.

Nous proposons finalement une nouvelle façon d'aborder le lissage dans la modélisation du graphe. En stéréo classique, le lissage proposé est toujours constant. Dans notre approche, nous tenons plutôt compte de la fonction de coût aux extrémités des arcs de lissage ainsi que de leur longueur pour établir leur capacité. Cette approche se base sur le principe qu'un arc saturé coupe la solution, et sur le fait qu'on peut calculer

d'éliminer l'étape de segmentation manuelle des images, ce qui réduirait encore les interventions de l'utilisateur requises.

5.4 Conclusion

En conclusion, nous pouvons dire que l'algorithme du flot maximum dans un graphe se généralise bien de la stéréoscopie classique à la reconstruction volumétrique. Il reste évidemment beaucoup de travail à faire avant d'avoir une méthode qui remplace vraiment les techniques séquentielles comme le *space carving*, mais les résultats indiquent néanmoins qu'il s'agit d'une avenue intéressante pour la recherche future.

RÉFÉRENCES

[1] Peter N. Belhumeur. A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237–260, 1996.

[2] Stan Birchfield et Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

[3] Yuri Boykov, Olga Veksler, et Ramin Zabih. Fast approximate energy minimization via graph cuts. Dans *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 377–384, 1999.

[4] A. Broadhurst, T.W. Drummond, et R. Cipolla. A probabilistic framework for space carving. Dans *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 388–393, 2001.

[5] Boris V. Cherkassky. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[6] Boris V. Cherkassky et Andrew V. Goldberg. On implementing push-relabel method for the maximum flow problem. Rapport Technique STAN-CS-94-1523, Department of Computer Science, Stanford University, 1994.

[7] Kong Man Cheung, Takeo Kanade, J.-Y. Bouguet, et M. Holler. A real time system for robust 3d voxel reconstruction of human motions. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 714 – 720, 2000.

[8] C. H. Chien et J. K. Aggarwal. Volume / surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 36:100–113, 1986.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, et Clifford Stein. *Introduction à l'algorithmique.* Dunod, 2002.

[10] Ingemar J. Cox, Sunita L. Hingorani, Satish B. Rao, et Bruce M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[11] Franklin C. Crow. Summed-area tables for texture mapping. Dans *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212. ACM Press, 1984.

[12] W. Bruce Culbertson, Thomas Malzbender, et Gregory G. Slabaugh. Generalized voxel coloring. Dans *Lecture Notes in Computer Science*, volume 1883, pages 100–115, 1999.

[13] Jeremy S. DeBonet et Paul Viola. Roxels: responsibility weighted 3d volume reconstruction. Dans *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 418–425, 1999.

[14] Olivier Faugeras et Renaud Keriven. Complete dense stereovision using level set methods. *Lecture Notes in Computer Science*, 1406:379–393, 1998.

[15] J.D. Foley, A. van Dam, S.K. Feiner, et J.F. Hughes. *Computer Graphics : Principles and Practice, 2e édition.* Addison-Wesley, 1990.

[16] Andrew V. Goldberg. http://www.avglab.com/andrew/soft.html, valide en date du 15-12-2004.

[17] Sing Bing Kang, Richard Szeliski, et Jinxiang Chai. Handling occlusions in dense multi-view stereo. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 103–110, 2001.

[18] Vladimir Kolmogorov et Ramin Zabih. Multi-camera scene reconstruction via graph cuts. Dans *European Conference on Computer Vision*, pages 82–96, 2002.

[19] Jaroslav Krivanek, Sumanta Pattanaik, et Jiri Zara. Adaptive mesh subdivision for precomputed radiance transfer. Dans *Spring Conference on Computer Graphics*, pages 106–111, 2004.

[20] Kiriakos N. Kutulakos. Approximate n-view stereo. Dans *European Conference on Computer Vision*, pages 67–83, 2000.

[21] Kiriakos N. Kutulakos et Steven M. Seitz. A theory of shape by space carving. Rapport Technique TR692, Comp. Science Department, U. Rochester, 1998.

[22] William E. Lorensen et Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[23] D. Marr et T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[24] Worthy N. Martin et J.K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[25] Yuichi Nakamura, Tomohiko Matsuura, Kiyohide Satoh, et Yuichi Ohta. Occlusion detectable stereo: Occlusion patterns in camera matrix. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[26] Stanley J. Osher et Ronald P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer Verlag, 2002.

[27] Andrew C. Prock et Charles R. Dyer. Towards real-time voxel coloring. Dans *Proc. Image Understanding Workshop*, pages 315–321, 1998.

[28] Sébastien Roy et Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. Dans *Proceedings of the IEEE International Conference on Computer Vision*, pages 492–502, 1998.

[29] Dave Rusin, 1998. http://www.math.niu.edu/~rusin/known-math/95/sphere.faq, valide en date du 15-12-2004.

[30] Stuart J. Russell et Peter Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, 1995.

[31] D. Scharstein, R. Szeliski, et R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[32] Daniel Scharstein et Richard Szeliski. http://cat.middlebury.edu/stereo/, valide en date du 19-03-2005.

[33] S. Seitz et C. Dyer. Photorealistic scene reconstruction by voxel coloring. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–1073, 1997.

[34] D. Snow, P. Viola, et R. Zabih. Exact voxel occupancy with graph cuts. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 345–353, 2000.

[35] Jian Sun, Nan-Ning Zheng, et Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[36] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.

[37] Emanuele Trucco et Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[38] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[39] Olga Veksler. Fast variable window for stereo correspondence using integral images. Dans *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 18–20, 2003.

[40] George Vogiatzis, Philip Torr, Steven M. Seitz, et Roberto Cipolla. Reconstructing relief surfaces. Dans *British Machine Vision Conference*, pages 117–126, 2004.

[41] Shin Yoshizawa, Alexander G. Belyaev, et Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. Dans *ACM symposium on Solid modeling and applications*, pages 247–253, 2003.

[42] C. Lawrence Zitnick et Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):1–10, 2000.