Université de Montréal

Reconstruction interactive de modèles hiérarchiques par sculpture d'espace

par

Martin Granger-Piché

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

Décembre 2004

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Reconstruction interactive de modèles hiérarchiques par sculpture d'espace

présenté par

Martin Granger-Piché

a été évalué par un jury composé des personnes suivantes :

Président:	Victor Ostromoukhov
Directeur de recherche:	Pierre Poulin
Membre:	Max Mignotte

Sommaire

De plus en plus d'applications dans les domaines des jeux vidéo, du cinéma, de la réalité virtuelle, etc., demandent des modèles 3D de qualité et d'apparence réaliste. Plusieurs systèmes de reconstruction automatiques existent, malheureusement certains objets complexes peuvent difficilement être numérisés avec ces systèmes. Une alternative consiste à intégrer l'usager dans le processus de reconstruction pour aider à resoudre certains problèmes des algorithmes. Dans ce contexte, nous présentons un système de reconstruction interactif basé sur la sculpture d'espace. L'usager contrôle la caméra et la position de l'objet et leur impact sur la reconstruction est immédiatement affiché à l'écran grâce à un affichage hardware efficace. Les nombreuses images passant dans le système sont d'abord nettoyées avec des ellipsoïdes de couleurs. Ensuite elles sont utilisées pour la sculpture des silhouettes. Finalement les images les plus représentatives sont gardées en mémoire dans une structure hémisphérique balancée pour la sculpture d'espace basée sur la comparaison de couleurs. Des points de couleur de calibration sont projetés sur des positions connues de l'environnement de reconstruction ou directement sur l'objet (ces points sont acquis par lumière structurée) pour permettre une calibration de la caméra automatique et adaptative. La structure hiérarchique (octree) est centrale dans notre algorithme de sculpture d'espace : les voxels sont sculptés efficacement (visibilité des voxels en hardware) à leur plus grandes tailles d'abord, les images sont traitées au bon niveau de détails de la structure et les voxels sont subdivisés uniquement à la surface du modèle 3D. Nous concluons avec une analyse de nos résultats et nous discutons de notre expérience acquise.

Mots clefs : *Reconstruction à partir d'images, système interactif, sculpture d'espace hiérarchique, projecteur, zoom, points de calibration.*

Abstract

Computer graphics is ubiquitous in computer entertainment, computer games, movie special effects, e-commerce, training in virtual environments, etc. The explosion of computer graphics applications fosters a growing need for accurate and realistic 3D models of real objects. Cheaper and flexible scanning devices that are adapted to specific needs and conditions must respond to this challenging demand. We present an interactive reconstruction system based on space carving. The user controls the camera and object positions and their impact on the reconstruction is immediately displayed with an efficient hardware display. The flow of images is cleaned-up with color ellipsoids, then is used for silhouette carving, and, finally, representative images are stored in a hemispherical structure for balanced color carving. Projected color points on the measured background or on the object (acquired by structured light) allow for automatic and adapted calibration of the camera. The octree object representation is central to our space carving algorithm : voxels are efficiently carved out at their largest size (hardware voxel visibility), images are treated at their appropriate level, and 3D regions are subdivided only when necessary. We conclude by analyzing our results and by discussing our acquired experience.

Key words : *Image-based reconstruction, interactive system, octree space carving, projector, zooming, calibration points.*

Table des matières

Liste des tableaux

Table des figures

Remerciements

Plusieurs personnes extraordinaires méritent sincèrement d'être remerciées pour m'avoir permis de cheminer autant dans mes études que dans ma vie et pour avoir cru en moi.

Dans un premier temps, j'aimerais remercier du fond du coeur Pierre Poulin qui a investi beaucoup de temps à me former à faire de la recherche, des présentations et qui m'a transmis sa passion de l'infographie. J'espère pouvoir continuer pendant longtemps notre belle collaboration.

Mes parents, en plus de m'avoir soutenu psychologiquement et financièrement pendant ces longues années d'études, m'ont transmis ce que je considère comme le plus beau cadeau : vivre passionnément tout ce que j'entreprends. Denise et Victor, je ne vous remercierai jamais assez !

Merci à Emric, mon meilleur ami avec qui je n'ai cessé de travailler en équipe, Caroline qui met du bonheur dans ma vie à tous les jours, Jacinthe et Jasmin pour tout leur support et à qui je dois mon intrusion dans l'informatique, Katerine et tous mes amis du DIRO, Mathieu pour les discussions enflammées sur n'importe quel sujets, tous les membres du LIGUM et tout le monde qui sont présents pour moi.

J'aimerais aussi remercier Victor Ostromoukhov, président de mon jury et qui, dans le cadre du cours IFT3051, m'a ouvert de grandes portes sur le monde de l'infographie et Max Mignotte membre du jury. Merci finalement au FCAR équipe, au MITACS et au CRSNG pour leur appui financier à mes études.

Chapitre 1

Introduction

L'infographie occupe une place indiscutable dans l'industrie du divertissement, des jeux vidéo, des effets spéciaux au cinéma, du commerce électronique, des environnements virtuels, etc. Les avancées technologiques des cartes graphiques dans les dernières années ainsi que de la puissance de calcul et de la taille de la mémoire à des coûts raisonnables ont ouvert la voie au développement d'une nouvelle variété d'applications. Cette explosion d'applications graphiques a entraîné un besoin grandissant pour des modèles 3D précis et d'apparence réaliste à partir d'objets réels. Des systèmes de reconstruction 3D moins dispendieux, plus flexibles, adaptés à des besoins précis et à des conditions particulières devraient répondre à une telle demande.

1.1 Philosophie de notre système

Notre objectif est d'obtenir des reconstructions 3D de qualité avec un équipement abordable pour le grand public et à l'intérieur d'un système flexible. La flexibilité est réalisée grâce à une série d'outils mis en place afin de permettre à l'usager de guider l'algorithme de reconstruction pendant son exécution. L'interactivité résulte d'un retour visuel en temps réel et d'une prise d'images simultanée, et montre à l'usager l'impact des dernières images acquises sur le modèle 3D en reconstruction. De plus, les outils ont été développés de manière à rendre la tâche de reconstruction simple et plus agréable pour l'usager. La Figure 1.1 schématise notre boucle de reconstruction incluant les différentes étapes d'une séance de reconstruction.

Il y a des avantages et des inconvénients à intégrer l'usager dans le processus de

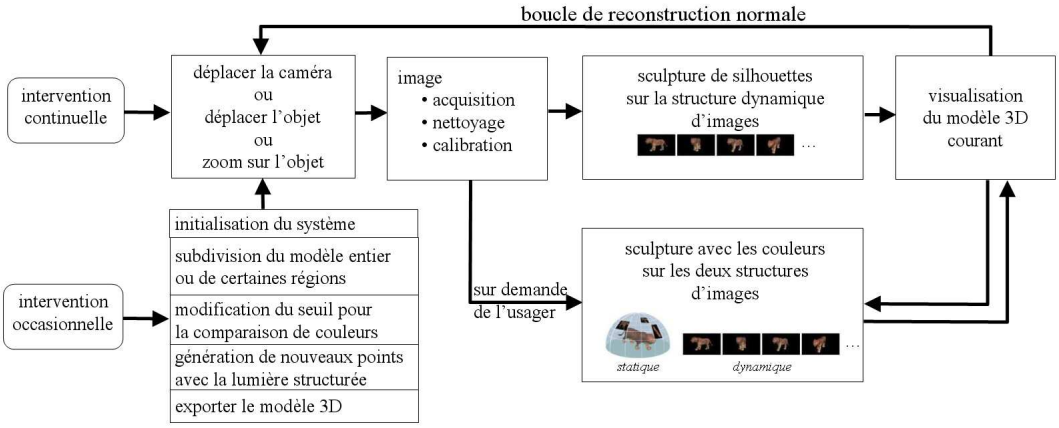

FIG. 1.1 – Boucle de reconstruction.

reconstruction. Un désavantage est qu'il est plus coûteux en temps pour l'usager d'être présent et d'intervenir durant la reconstruction par rapport à un système automatique. Par contre, certains objets ayant une géométrie complexe, des surfaces spéculaires, etc., peuvent poser des problèmes à des systèmes automatiques. C'est dans ce contexte que l'usager peut intervenir dans le processus de reconstruction pour guider la reconstruction soit en ajoutant de nouvelles images aux endroits appropriés, soit en contrôlant des variables de système pour s'adapter aux différents types d'objets, soit en ajoutant des images en gros plan sur l'objet pour procurer plus de détails, etc. Le système que nous proposons a été développé avec cette philosophie afin de vérifier à quel point l'usager peut améliorer la qualité des modèles reconstruits en utilisant les différents outils interactifs mis en place dans notre système.

1.2 Outils mis en place pour l'interactivité

Nous avons observé qu'une des parties coûteuses en temps et en effort des algorithmes de reconstruction est la capture, la calibration et la segmentation des images. Parfois, des heures peuvent être consommées uniquement à ces tâches. Ceci constitue une limitation importante lorsque vient le temps de reconstruire plusieurs objets. De plus, il peut être décevant pour l'usager de se rendre compte à la fin du processus de reconstruction qu'il aurait dû ajouter plus d'images de certains angles pour obtenir les résultats escomptés et qu'il doit malheureusement recommencer depuis le début. Les outils mis en place

dans notre système tentent de répondre à ces difficultés :

1. calibration automatique de la caméra ;
2. estimation automatique de la pose de l'objet ;
3. calibration avec points de couleur projetés directement sur la surface de l'objet pour l'obtention de détails de surface ;
4. segmentation automatique de l'arrière-plan dans les images ;
5. visualisation de la reconstruction en temps réel ;
6. structure hiérarchique (octree) pour gérer les différents niveaux de détail et accélérer la sculpture d'espace ;
7. structures internes d'images facilitant le grand nombre d'images traitées dans le système.

L'intégration d'un projecteur dans le système facilite l'exécution de plusieurs des outils mentionnés ci-haut. Il sert, entre autres, à illuminer des positions 3D connues pour faciliter la calibration automatique de la caméra et le suivi automatique de l'objet. De plus, il permet d'obtenir des points 3D sur la surface de l'objet afin de calibrer des images en mode zoom procurant plus de détails (Section 4.1.4).

Nous soulignons, dans la section qui suit, les contributions apportées par notre recherche.

1.3 Contributions

La mise en oeuvre d'un système de reconstruction complet et interactif requiert le développement de plusieurs algorithmes. Certains proviennent d'articles référés dans ce mémoire, mais d'autres résultent de notre recherche. Les principales contributions de notre projet, regroupées par classe de tâches, sont les suivantes :

1. Système interactif
   - outils pour donner un contrôle à l'usager
   - boucle de retour visuel
   - acquisition/segmentation/calibration des images
   - affichage de la reconstruction en temps réel.
2. Calibrations à base d'un projecteur
   - calibration automatique de la caméra avec des points de couleur projetés
   - calibration en mode zoom avec des points projetés sur l'objet obtenus avec un algorithme de lumière structurée
   - estimation automatique de la pose de l'objet.
3. Sculpture d'espace hiérarchique (octree)
   - économie d'espace mémoire puisque seuls les voxels de la surface du modèle 3D sont subdivisés
   - efficacité en tentant d'éliminer les gros voxels avant d'être subdivisés
   - visibilité des voxels traitée avec la carte graphique.
4. Traitement d'images
   - ellipsoïdes RGB permettant une détection des couleurs robuste et rapide
   - segmentation automatique de l'arrière-plan.

Nos contributions par rapport à la structure hiérarchique sont au niveau de l'intégration de cette structure dans la sculpture par comparaison de couleurs ainsi que le traitement approprié des détails des images. Cette structure est aussi utilisée dans d'autres techniques de reconstruction. Par exemple, Pulli *et al.* [PDH+97] utilisent un octree comme structure pour une sculpture à partir d'images de profondeurs. Par contre, aucun test de comparaison de couleurs n'est effectué.

Les différentes contributions mentionnées ci-haut ont fait l'objet d'une publication à la conférence "*Vision, Modeling, and Visualization 2004*" qui se tenait à l'université Stanford [GPEP04].

1.4 Structure du mémoire

Le Chapitre 2 présente une brève revue de la littérature sur la reconstruction d'objets. Le Chapitre 3 dépeint le cheminement d'une session typique de reconstruction avec

notre système. Le Chapitre 4 entre dans les détails des techniques utilisées dans notre système. Le Chapitre 5 montre différents modèles 3D reconstruits avec notre système accompagnés de quelques statistiques et analyses sur ces derniers. Finalement, dans le Chapitre 6 se trouvent les conclusions de nos travaux ainsi qu'une ouverture sur les travaux futurs. En annexe D se trouvent des définitions de termes que nous utilisons fréquemment durant ce mémoire.

Chapitre 2

Revue de la littérature

Il existe beaucoup de techniques pour numériser des objets. Le problème de la modélisation et de la reconstruction 3D intéresse les chercheurs depuis longtemps dans les domaines de la vision 3D par ordinateur et de l'infographie. Dans ce chapitre, nous faisons un bref survol de certaines de ces techniques. Plus précisément, nous présentons la modélisation et le rendu à partir d'images, les techniques de reconstruction 3D actives et passives, la reconstruction interactive, et finalement la technique que nous utilisons, la sculpture d'espace. Nous concluons ce chapitre en soulignant nos contributions à l'intérieur de ces domaines.

2.1 Modélisation et rendu à partir d'images

Un des objectifs de l'infographie dans les dernières années est d'obtenir des scènes 3D toujours plus réalistes, respectant autant que possible les propriétés physiques de la propagation de la lumière (illumination globale), avec des animations de caractère plus fluides (capture de mouvements par senseurs), etc. Dans ces scènes, il y a toujours besoin de modèles 3D, qui se doivent d'être toujours plus près de la réalité. C'est dans ce contexte que plusieurs techniques ont été développées en infographie pour répondre à ce besoin.

La modélisation et le rendu à partir d'images (*image-based modeling and rendering*) tentent de répondre à cette réalité. Les contributions importantes dans ce domaine incluent le *light field* [LH96], le *lumigraph* [GGSC96, BBM+01], les *visual* et *opacity hulls* [MBR+00, MPN+02], etc. Ces techniques utilisent l'information des images d'un objet

pour synthétiser de nouvelles images à de nouveaux points de vue. Comme les nouvelles images sont construites à partir de vraies images de l'objet, le résultat est en général très réaliste et même les effets dépendant du point de vue, comme les spécularités, sont présents. Certaines de ces techniques [GGSC96, BBM+01, LH96] n'ont aucun modèle 3D pour supporter la génération de nouvelles images. Les *opacity hulls* [MPN+02] reposent sur un modèle 3D construit à partir des silhouettes extraites des images de l'objet. La qualité visuelle de ces modèles est impressionnante. La semi-transparence captée par cette technique permet de reconstruire plusieurs types d'objets. De plus, ses auteurs ont présenté une façon de ré-illuminer le modèle 3D. Le gros problème de cette technique est l'espace mémoire requis pour conserver l'information capturée, qui est de l'ordre de quelques GBs.

Même si la modélisation et le rendu à partir d'images ont su procurer des résultats visuels très plaisants, l'énorme espace mémoire requis, l'illumination incidente capturée avec le modèle ou l'équipement dispendieux requérant une calibration précise sont des limitations qui affectent l'adoption à grande échelle de ces techniques pour des systèmes d'acquisition 3D généraux.

2.2 Vision 3D par ordinateur

La vision par ordinateur traditionnelle a également développé plusieurs techniques passives et actives, comme la stéréo, le flux optique, la numérisation 3D laser, la lumière structurée, la sculpture d'espace, etc.

Les techniques passives (stéréo, sculpture d'espace, flux optique) tentent de reconstruire les objets à partir d'images calibrées de ceux-ci contenant l'objet sous différents angles de vue. Ces techniques trouvent les positions 3D les plus cohérentes, en se basant sur certains critères comme la couleur des pixels dans lesquels ces positions 3D se projettent. Dans la plupart des cas, une grille 3D régulière est utilisée pour faciliter la recherche des positions 3D valides. Par contre, d'autres techniques, comme POMO [PSD+03], utilisent des points 3D générés aléatoirement à l'intérieur d'une région définie.

La stéréo est une technique de reconstruction 3D classique [DA89]. Les caméras sont calibrées relativement entre elles et elles doivent se situer près l'une de l'autre. Les pixels des images sont mis automatiquement en correspondance par corrélation et la disparité des pixels (la profondeur) est retrouvée par triangulation. Le résultat de la reconstruc-

tion est une carte de profondeurs d'un point de vue. Le désavantage de la contrainte de proximité des caméras est qu'il n'est possible de reconstruire qu'une face d'un objet avec cette approche. Il est possible par contre d'obtenir plusieurs reconstructions de différents points de vue de l'objet et de fusionner les différentes faces reconstruites de l'objet en post-traitement. Une autre difficulté de cette approche est la mise en correspondance des pixels qui peut être affectée par le bruit dans les images, les occlusions, les propriétés de réflectance des surfaces de l'objet, etc.

Une autre approche, qui est d'un grand intérêt pour nous, est la reconstruction basée sur la photo-consistance [SD99, KS00]. Un modèle 3D est photo-consistant si pour chaque pixel dans lequel il se projette et pour chaque image, la couleur du pixel est "similaire" à la couleur de la projection du modèle 3D reconstruit. L'approche par coloration de voxels de Seitz et Dyer [SD99] restreint les points de vue à se situer à l'extérieur de l'objet à reconstruire. Kutulakos et Seitz [KS00] ont proposé une généralisation de cette technique avec la sculpture d'espace (*space carving*). Nous en discutons plus en détail dans la Section 2.4.

Ce qui distingue les techniques actives (numériseur 3D laser, lumière structurée, etc.) des techniques passives est l'utilisation d'une source active émettrice de lumière pour faciliter l'identification les positions 3D. Un projecteur ou un laser calibré relativement avec une caméra permet d'identifier des correspondances entre des pixels projetés par le projecteur et leurs positions retrouvées dans la caméra. La déprojection d'un pixel donne un vecteur en 3D. Une fois la correspondance établie entre un pixel de la caméra et un pixel du projecteur ou un laser, l'intersection en 3D des deux vecteurs donne la position 3D recherchée. Ces techniques permettent de reconstruire des modèles 3D ayant les meilleures précisions. Comme la stéréo, ces techniques permettent de reconstruire que des sections d'un objet "visibles" des deux dispositifs et par conséquent il est nécessaire d'appliquer plusieurs reconstructions et de les fusionner en post-traitement.

Plusieurs recueils discutent en détail les avantages et les inconvénients de ces techniques [BR00, CM99, SG99].

2.3 Reconstruction interactive

Le processus de reconstruction comporte en général un ensemble de tâches à accomplir dans un ordre séquentiel :

1. acquisition d'images/vidéo ;
2. calibration manuelle/automatique des images/*frames*, nettoyage des images, masques de silhouettes, etc. ;
3. initialisation des variables du système ;
4. reconstruction 3D ;
5. affichage du modèle 3D reconstruit.

Ce processus séquentiel, qui comporte des étapes longues sans aucun retour d'informations intermédiaires, est moins approprié pour reconstruire des modèles 3D réalistes adaptés à des besoins spécifiques.

Plusieurs systèmes de reconstruction académiques [DTM96, POF98, Ded01, PSD+03] et industriels [Rea, Can, Pho, Bou] ont reconnu cette limitation et ont permis des interventions de l'usager à différentes étapes du processus de reconstruction. Étant donné que l'usager est en mesure de comprendre la complexité de l'objet à reconstruire et les régions pouvant poser des problèmes à l'algorithme de reconstruction, il est mieux placé pour indiquer où ajouter des polygones, extraire des textures représentatives, éliminer des images inappropriées, etc. Aussi, avec des systèmes de reconstruction plus simples basés sur les points, l'usager peut spécifier interactivement où de nouveaux points sont nécessaires [Rea, RHHL02]. Ces techniques ont prouvé qu'elles pouvaient reconstruire des modèles 3D plus réalistes, mais elles demandent tout de même une implication relativement pénible de l'usager et chaque nouvelle image requiert plus d'efforts.

Rusinkiewicz *et al.* [RHHL02] ont marqué une avancée remarquable par rapport à ces systèmes interactifs. Dans leur système, l'usager manipule lentement l'objet réel à reconstruire en même temps qu'il visualise en temps réel les points 3D les plus robustes reconstruits par un algorithme de lumière structurée. Seulement des équipements abordables étaient considérés. Une fois l'usager satisfait des points reconstruits, un algorithme d'optimisation globale de type *Iterative Closest Point* (ICP) est appliqué sur l'ensemble des images préalablement acquises. Les modèles 3D découlant de cette optimisation sont de qualité comparable à des reconstructions par numériseur 3D laser. Malheureusement, leur système souffre des mêmes limitations traditionnelles de la lumière structurée : l'objet doit être principalement diffus et doit posséder peu de textures. Epstein [Eps05] propose d'utiliser des miroirs pour reconstruire plusieurs faces des objets simultanément, améliorant par conséquent la "couverture" de la lumière structurée,

mais sans toutefois en réduire ses limitations traditionnelles.

2.4 Sculpture d'espace

Nous avons choisi la sculpture d'espace (*space carving*) [SD99, KS00] comme technique de reconstruction parce que nous estimons qu'elle a produit jusqu'à maintenant parmi les meilleurs résultats. Dans la sculpture d'espace, les voxels d'une grille 3D régulière sont projetés avec cohérence dans des images calibrées. Les couleurs de chaque voxel sont extraites à partir de chaque image dans lesquelles il est visible. Les voxels avec des couleurs *non-consistentes* ou se projetant à l'extérieur de l'objet sont éliminés de l'ensemble de voxels. Les voxels colorés restants forment le modèle 3D reconstruit.

La photo-consistance se base en général sur le fait que la couleur d'un point de surface diffuse reste invariante aux changements de points de vue. Donc, un voxel valide possède la même couleur (ou "similaire" si on considère le bruit dans les images) dans toutes les images dans lesquelles il est visible. La comparaison de couleurs consiste par conséquent à calculer la variance des couleurs des pixels visibles pour un voxel donné et de le rejeter si cette variance excède un certain seuil.

La cohérence de la grille 3D régulière permet une détermination efficace de la visibilité. Les voxels sont traités par tranches (un plan) successives suivant les axes du volume englobant l'objet. Dans un premier temps, les tranches de voxels perpendiculaires à l'axe des x sont projetées dans les images visibles d'une extrémité du volume englobant à l'autre. Chaque voxel ne respectant pas le test de photo-consistance est éliminé. La visibilité est traitée par coloration. Si un voxel est valide, il laisse sa trace dans les pixels des images dans lesquelles il est visible. Tous les voxels des tranches suivantes se projetant dans ces pixels colorés sont considérés cachés. Ceci constitue une itération. Ce processus est répété, avec les voxels valides restant des itérations précédentes, en suivant les trois axes (X, Y, Z) et dans les deux directions de chaque axe. Les voxels restants forment le modèle 3D reconstruit. La Figure 2.1 illustre ce procédé. L'aspect conservateur de la comparaison de couleurs (même pour des surfaces non diffuses [YPW03]) assure qu'à chaque étape, seulement des voxels impossibles peuvent être éliminés.

La sculpture d'espace est plus efficace pour des surfaces texturées. Il est possible d'augmenter son efficacité en utilisant l'information des silhouettes pour éliminer de grandes portions d'espace extérieures à l'objet. La Figure 2.2 montre un résultat tiré de

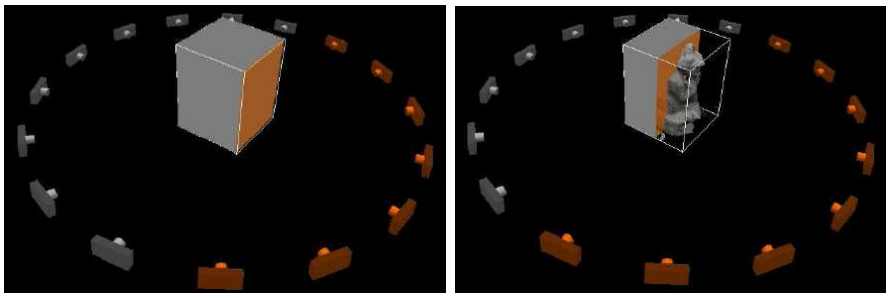

FIG. 2.1 – Traitement de la visibilité par plan suivant les axes du volume englobant l'objet dans la sculpture d'espace telle que présenté dans Kutulakos et Seitz [KS00]. Image tirées de [KS00].

Kutulakos et Seitz [KS00] obtenu par cette technique. Il a fallu à peu près 110 minutes pour cette reconstruction. La résolution du modèle est de $256 \times 256 \times 256$.

Dans le prochain chapitre, nous décrivons les différentes parties d'une session de reconstruction avec notre système afin d'aider à mieux comprendre la description des différentes techniques que nous utilisons dans notre système (Chapitre 4).

FIG. 2.2 – Résultat obtenu par Kutulakos et Seitz [KS00]. Images tirées de [KS00].

Chapitre 3

Séance de reconstruction

FIG. 3.1 – Shéma de notre système de reconstruction.

Dans ce chapitre, nous présentons une séance typique de reconstruction pour donner une vue globale de l'utilisation du système. Les chapitres suivants couvrent en détail les différents aspects algorithmiques de notre système.

Notre système de reconstruction consiste en une installation simple composée d'équipements relativement abordables : un ordinateur contrôlant une caméra vidéo digitale (connectée sur le port *firewire*), un projecteur DLP, un environnement de reconstruction de dimensions connues (trois panneaux peints en blanc mat) sur lequel repose l'objet à reconstruire. L'objet peut être placé sur un piédestal muni de marqueurs réels de couleur, permettant éventuellement de le déplacer. La Figure 3.1 donne une vue schématique de l'installation. La Figure 3.2 montre le cheminement entre les étapes du processus de reconstruction, que nous décrivons dans les sections qui suivent.

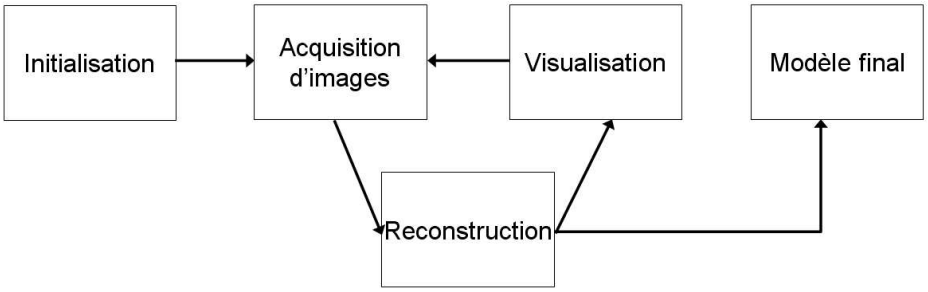

FIG. 3.2 – Processus de reconstruction.

Initialisation

Dans un premier temps, le projecteur est calibré manuellement (Section 4.1.1) par rapport à l'environnement de reconstruction. Il sera utilisé dans les étapes ultérieures afin de faciliter l'acquisition de différentes informations par la caméra.

Plusieurs traitements basés sur la couleur sont effectués sur les images acquises lors du processus de reconstruction (segmentation de l'objet, identification des points de calibration, suivi du piédestal, etc.). Une analyse de la réponse des couleurs de la caméra est donc effectuée en approximant les différentes couleurs utilisées par des ellipsoïdes RGB (Section 4.4.1). Cet outil (les ellipsoïdes) s'est avéré très efficace en rapidité et en précision pour détecter si une couleur appartient ou non à un groupe de couleurs donné.

Finalement, l'usager doit spécifier la taille du cube englobant l'objet à reconstruire. Il est important que l'objet soit complètement compris à l'intérieur de ce cube, sinon certaines parties de l'objet pourraient ne pas être reconstruites. Afin d'aider l'usager à choisir la bonne taille, le projecteur calibré projette le cube englobant avec la taille choisie à l'intérieur de l'environnement de reconstruction (Figure 4.3 droite).

Acquisition d'images

Une fois le système initialisé, l'usager déplace manuellement la caméra vidéo autour de l'objet. Le projecteur calibré projette des points de calibration de couleurs différentes sur les plans de l'environnement de reconstruction de manière à calibrer automatiquement les nouvelles images acquises par la caméra (Section 4.1.2). Ces images sont automatiquement nettoyées (Section 4.4) et ajoutées aux images utilisées pour la reconstruction. De manière à traiter efficacement la grande quantité d'images entrant dans le système, deux structures, une statique et une dynamique (FIFO), sont introduites (Section 4.2.1).

Étant donné que l'installation limite les angles de vue de la caméra, l'objet peut être placé sur un piédestal. Il peut ainsi être déplacé et sa position suivie automatiquement pour ensuite continuer l'acquisition d'images (Section 4.1.3).

Reconstruction

L'algorithme de reconstruction utilise les images disponibles dans le système. La sculpture des silhouettes (Section 4.2.3) contribue beaucoup à l'aspect interactif du système puisque le rejet d'un voxel se projetant à l'extérieur de l'objet est très rapide. Le rejet basé uniquement sur les couleurs (Section 4.2.4) nécessite de recueillir des couleurs dans un ensemble d'images avant de pouvoir être effectué, ce qui représente un processus plus long.

Visualisation de la reconstruction

Les voxels non rejetés sont affichés (Section 4.5) durant l'acquisition d'images. Grâce à ce retour visuel d'information, l'usager peut concentrer les nouveaux points de vue aux endroits appropriés. L'affichage est fait en OpenGL et les paramètres de la caméra virtuelle correspondent à ceux de la calibration de la dernière image acquise. Ceci permet à l'usager d'observer directement l'influence des images qu'il acquiert de l'objet sur la reconstruction du modèle 3D.

Structure hiérarchique

Une structure hiérarchique (octree) des voxels permet de choisir le niveau de détail approprié en fonction des images entrantes. Cette structure permet d'économiser de l'espace mémoire en plus d'accélérer la reconstruction puisque de gros voxels peuvent être découpés avec les silhouettes avant d'être subdivisés (Section 4.3).

Modèle final

Lorsque le modèle reconstruit a atteint la qualité désirée, il est possible de l'exporter sous forme de points avec une couleur et une normale au niveau de subdivision voulu. Ce modèle est compatible avec *QSplat* [RL00], une application permettant d'afficher avec qualité un grand nombre de points en temps réel sous forme de *splatting* (Section 4.6).

Chapitre 4

Description du système

Nous décrivons plus en profondeur les différentes parties de notre système dans ce chapitre. Nous discutons des aspects de calibration, des particularités de notre implémentation de la sculpture d'espace, du choix d'une structure hiérarchique de voxels, des traitements effectués sur les images, des différents modes d'affichage et finalement de l'exportation du modèle géométrique vers d'autres applications d'affichage de modèles 3D.

Le chapitre suivant discutera et analysera en détail les résultats obtenus.

4.1 Calibration

La calibration de caméra consiste à établir la correspondance entre des points du monde 3D et leur projection dans le monde 2D. À l'aide de ces correspondances, il est possible d'extraire les paramètres intrinsèques et extrinsèques de la caméra. Les paramètres intrinsèques d'une caméra sont la distance focale de la lentille, le centre de projection, la taille de l'image et la résolution (nombre de pixels). Les paramètres extrinsèques sont la position et l'orientation de la caméra par rapport à un système de coordonnées. La littérature sur la calibration de caméra est vaste [Fau93, TV98, Aya91, KSK98, Tsa87, DTM96, GGSC96, Ded01]. Certaines de ces techniques supposent que les paramètres intrinsèques de la caméra sont connus, ce qui permet de contraindre davantage le système d'équations à résoudre. Par conséquent, moins de correspondances 3D-2D sont nécessaires, i.e. jusqu'à un minimum de 4 correspondances. D'autres techniques [Fau93, Ras03a] permettent de retrouver les paramètres intrinsèques et extrinsèques au coût de plus de correspondances 3D-2D, i.e. au moins 8 correspondances. Nous avons

choisi la deuxième stratégie afin de permettre une plus grande flexibilité dans la prise d'images, par exemple pour laisser l'usager *zoomer* sur l'objet sans avoir à approcher physiquement la caméra.

Nous avons choisi d'extraire une projection affine plutôt qu'une projection perspective parce qu'elle permet de représenter des modèles de caméra plus généraux. Ce choix est motivé par le fait que le modèle de calibration d'un projecteur (Section 4.1.1) s'exprime mieux par une projection affine puisque le centre de projection du projecteur (supposé connu dans la calibration perspective) est décentré par rapport à l'axe vertical. Après plusieurs expérimentations, ce modèle s'est avéré beaucoup plus stable et précis. Nous obtenons une erreur moyenne de reprojection inférieure à 1 pixel alors que nous avions des erreurs de l'ordre de 15 pixels avec le modèle perspective [Fau93]. Étant donné que le projecteur a une résolution de 1024×768 pixels et que nous utilisons en moyenne 25 points de calibration, une erreur moyenne de 1 pixel représente 0.003% d'erreur alors qu'une erreur moyenne de 15 pixels représente 0.045% d'erreur.

Puisque nous travaillons avec OpenGL pour afficher le modèle 3D reconstruit selon la caméra calibrée, il est nécessaire de convertir la matrice de projection affine en matrices compatibles avec OpenGL (GL_PROJECTION et GL_MODELVIEW). Pour ce faire, nous nous sommes inspirés d'information mise disponible sur l'internet [Ras03b] mais qui semble désormais disparue. Nous avons mis en Annexe A le code C++ pour construire le système d'équations linéaire et pour le résoudre afin d'obtenir la projection affine. En Annexe B se trouve le code C++ pour obtenir la conversion de la matrice affine vers les matrices OpenGL.

Lors de la calibration automatique de la caméra, il existe une certaine imprécision sur la détection de la position des points de calibration dans l'image entraînant une plus grande erreur de calibration comparativement à une calibration manuelle. Pour remédier en partie à ce problème, nous perturbons aléatoirement les positions retrouvées dans l'image à l'intérieur d'un rayon d'environ 1 pixel et nous conservons la meilleure solution parmi environ 400 perturbations. Nous arrivons de cette manière à obtenir une erreur moyenne de reprojection de l'ordre de 1 pixel. Nous avons également constaté que la calibration était plus stable avec 10 correspondances 2D-3D et plus.

4.1.1 Calibration du projecteur

Un projecteur se comporte exactement comme un caméra à l'exception que la "projection" s'effectue dans le sens inverse. Dans une caméra, l'univers 3D se projette sur une image 2D alors qu'avec un projecteur, une image 2D se projette sur l'univers 3D. Ceci explique pourquoi il est possible de calibrer un projecteur avec la même technique qu'une caméra.

La calibration du projecteur s'effectue manuellement. L'environnement de reconstruction sert de système d'axes de référence. Des petites croix sont tracées sur l'environnement de reconstruction à des positions connues. L'usager identifie les correspondances 2D-3D en indiquant, à l'aide de la souris, les pixels du projecteur qui illuminent ces positions connues (Figure 4.1).

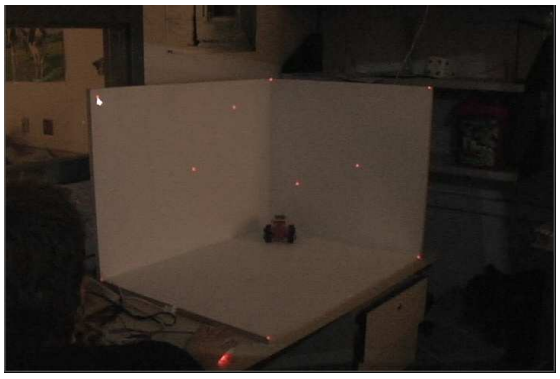

FIG. 4.1 – Calibration manuelle du projecteur : l'usager illumine avec des pixels rouges du projecteur les points de calibration mesurés dans l'environnement de reconstruction avec l'aide de la souris.

Une fois la calibration du projecteur terminée, il est possible d'identifier l'intersection entre la projection de n'importe quel pixel avec l'environnement de reconstruction.

4.1.2 Calibration automatique de la caméra

Étant donné que le projecteur est calibré et que nous connaissons la géométrie de l'environnement de reconstruction, il est possible d'illuminer, à l'aide du projecteur, des points dont nous connaissons la position et de les identifier automatiquement dans la caméra. L'idée originale était de projeter des points de couleur unique afin de permettre une correspondance 2D-3D dans chaque image. Malheureusement, après une étude de la réponse des couleurs dans la caméra, nous avons constaté que seulement les couleurs

saturées rouge, vert et bleu, et le gris permettaient une identification unique, robuste et sans fausses détections. Comme nous avons besoin de 12 points (8 en réalité mais nous voulons supporter la perte de certains points pour une calibration plus robuste), nous avons développé un encodage permettant d'utiliser plusieurs fois chaque couleur tout en conservant une identification unique de chaque point (Figure 4.2). Nous supposons que le *up vector* de la caméra reste toujours vers le haut. Avec cette hypothèse, nous projetons deux fois chaque couleur (rouge, vert, bleu) de telle sorte que dans l'image de la caméra il soit possible d'identifier chacun des points de la même couleur en fonction de l'axe vertical. Un point gris est projeté entre certaines paires de points de couleur. Chaque point gris est identifié uniquement une fois la position détectée des points de couleur sur son segment de droite. Nous éliminons les points de couleur détectés couvrant moins de 10 pixels dans la caméra, ce qui réduit les fausses identifications.

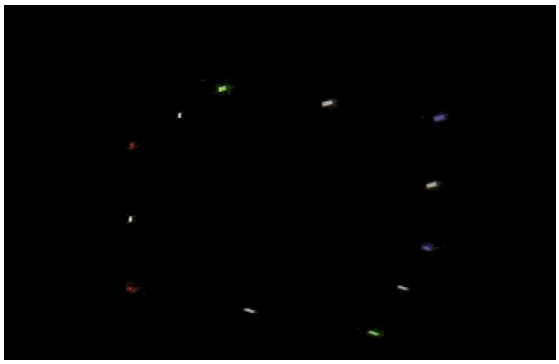

FIG. 4.2 – Disposition unique des points de calibration et de leurs couleurs.

Nous avons deux modes de calibration automatique de la caméra : le mode sécuritaire et le mode temporel.

Dans le mode sécuritaire (Figure 4.3 gauche), la zone où se trouve l'objet n'est pas éclairée par le projecteur, seuls les points de calibration sont illuminés. Ainsi, l'image est facilement nettoyée (Section 4.4) de telle sorte que chaque point puisse être identifié à l'aide des ellipsoïdes RGB (Section 4.4.1). Si la détection mène à une bonne calibration (erreur de calibration inférieure à 2 pixels), la position image de chacun des points de calibration est conservée et le mode temporel est activé.

Dans le mode temporel, la lumière illuminant l'objet est activée (Figure 4.3 droite) et chaque point de calibration est suivi à l'intérieur d'une fenêtre de 40×40 pixels par rapport à sa dernière position détectée. Lorsque l'erreur de calibration est inférieure à un

seuil (2 pixels), l'image est ajoutée dans le système. Dans le cas contraire, nous retournons automatiquement dans le mode sécuritaire. En général, une erreur de calibration élevée signifie qu'il n'existe pas de modèle de caméra pouvant bien expliquer les correspondances 2D-3D. Cela se produit lorsque certaines positions 2D ne sont pas détectées à l'endroit approprié. Dans notre contexte, un mouvement de caméra trop rapide ou des points de calibration cachés par l'objet, hors du champ de vue de la caméra ou mal détectés lors du nettoyage de l'image entraînent des erreurs de calibration élevées. En retournant dans le mode sécuritaire, tous les points peuvent être à nouveau identifiés uniquement. Ce processus permet une calibration automatique stable où l'usager n'a jamais à intervenir.

La distance entre les points de calibration et la zone éclairée par le projecteur peut être changée par l'usager. Si les points sont trop près, la position de certains points de calibration peut être mal détectée. Nous avons testé la possibilité de déterminer une distance image minimum à respecter et adapter automatiquement la position 3D des points projetés afin de toujours respecter cette distance. Malheureusement, étant donné le manque de synchronisation entre la caméra et le projecteur, il s'est avéré difficile d'identifier les bonnes positions 3D des points détectés dans une image. Il existe une latence d'environ 300 millisecondes entre la capture d'une image et sa réception dans notre programme. Ceci implique que si les positions des points 3D sont modifiées à une fréquence plus rapide que cette latence et surtout à une fréquence indéterminée, il est difficile de déterminer lors de la réception d'une image quelle était la configuration réelle des points de calibration au moment de sa capture. Nous avons donc décidé de laisser les positions des points de calibration fixes. Par contre, il est clair que dans le cas d'une synchronisation possible, cette solution permettrait une calibration automatique encore plus stable puisque les positions des points seraient mieux distribuées dans l'image de la caméra.

4.1.3 Estimation de la pose du piédestal

L'objet peut être placé sur un piédestal comportant des marqueurs réels de couleur. Initialement, l'usager identifie grossièrement le centre des marqueurs dans l'image de la caméra calibrée. Le système déplace automatiquement ces points pour qu'ils se situent le plus possible au centre de l'image dans la caméra de chaque marqueur, couvrant

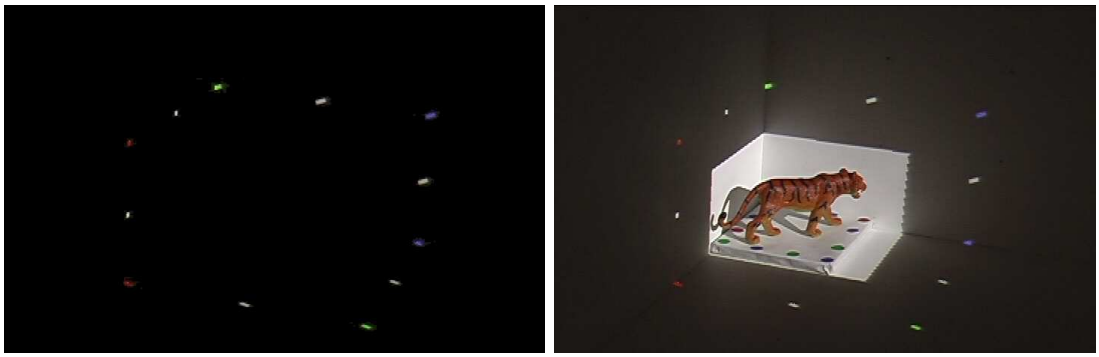

FIG. 4.3 – Gauche : image de la caméra dans le mode de calibration sécuritaire où seulement les points de calibration sont visibles. Le reste des pixels ont été nettoyés. Droite : image de la caméra dans le mode temporel. Entre les points apparaît l'objet éclairé par le projecteur. L'image a été nettoyée avec les ellipsoïdes d'arrière-plan et d'interréflexion (Section 4.4, Figure 4.12). Les points de calibration sont suivis temporellement à l'intérieur d'une fenêtre de 40×40 pixels.

environ 7×7 pixels chacun. Lorsque le système entre dans le mode de suivi de l'objet, l'illumination du projecteur est éteinte pour que chacun des marqueurs puisse être suivi temporellement à l'intérieur d'une fenêtre 6×6 pixels illuminée en blanc par le projecteur (Figure 4.4). Les marqueurs identifiés dans l'image de la caméra sont utilisés

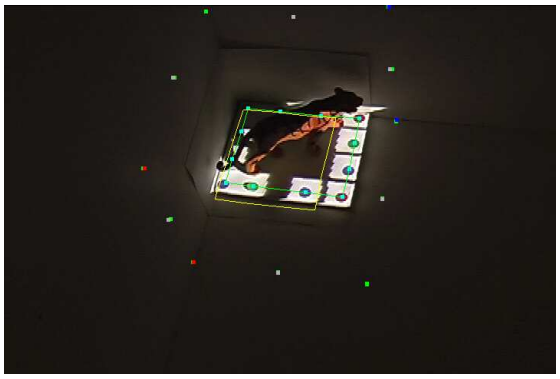

FIG. 4.4 – Les petites fenêtres d'illumination sur les marqueurs aident le suivi temporel de ces derniers. Le rectangle pointillé vert montre la position actuelle retrouvée alors que le rectangle jaune pointillé montre la position originale de l'objet (avant qu'il soit déplacé par l'usager).

pour retrouver leurs positions 3D correspondantes, ce qui permet l'extraction d'une transformation rigide (translation horizontale et rotation) à l'aide d'une approche par

moindres carrés [PFTV92]. Les détails de cette technique se trouvent dans le mémoire de Epstein [Eps05].

Cette approche nous permet de déplacer l'objet tout en connaissant la transformation pour le ramener à sa configuration initiale. En appliquant cette transformation sur la calibration de la caméra, nous pouvons palier au problème des angles de vue limités de notre système en allant chercher des points de vue de tous les côtés de l'objet.

Comme l'illumination de l'objet change en le tournant, il est donc important de bien la traiter dans l'algorithme de sculpture d'espace puisqu'une des hypothèses faites pour la comparaison de couleurs est une illumination fixe. Cet aspect sera discuté dans la Section 4.2.

4.1.4 Calibration en mode zoom

Si l'usager du système veut obtenir des détails fins d'une petite région de l'objet, il doit approcher la caméra sur cette région. Les points de calibration restreignent par contre cette possibilité puisqu'ils doivent toujours être visibles dans l'image de la caméra. Afin de remédier à ce problème, de nouveaux points de calibration sont nécessaires, mais cette fois ces points seront générés directement sur l'objet.

Nous utilisons la lumière structurée [RCM$^+$01, Eps05] afin de générer des points 3D sur la surface de l'objet à reconstruire. Dans un premier temps, la caméra est calibrée avec les points habituels de calibration. Ensuite, un groupe de points 3D générés par lumière structurée sont testés pour leur robustesse selon la position courante calibrée de la caméra. Nous choisissons parmi ces points seulement ceux dont la position détectée dans l'image est proche de celle espérée par notre calibration et qui sont suffisamment distants les uns par rapport aux autres pour réduire toute confusion dans leur détection ultérieure. Lorsqu'un nombre suffisant de ces points sont sélectionnés et qu'ils permettent une calibration équivalente à la calibration normale, nous pouvons utiliser ces points pour *zoomer* sur l'objet tout en obtenant une bonne calibration (Figure 4.5). Étant donné que nous utilisons plus de points que nécessaire, nous pouvons supporter le fait que certains points disparaissent dû à des occlusions ou qu'ils sortent de l'image lorsque la caméra effectue un zoom sur l'objet. Lorsque la caméra s'éloigne de la région où les points ont été validés, il se peut que le sous-ensemble choisi ne réponde plus au critère de proximité dans l'image. Dans ce cas, il est possible de sélectionner un nouveau

sous-ensemble de points parmi ceux initialement générés. Afin de retrouver ces points, la caméra doit être calibrée de nouveau avec les points du mode normal de calibration.

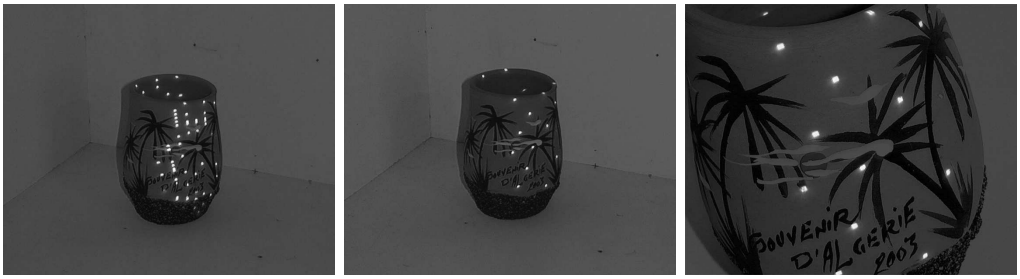

FIG. 4.5 – Gauche : un ensemble de points acquis avec la lumière structurée sont conservés comme candidats pour de nouveaux points de calibration. Centre : un sous-ensemble de points choisis pour leur robustesse et leur disposition en fonction de la position actuelle de la caméra. Droite : image calibrée en mode zoom avec ces points de calibration.

4.2 Sculpture d'espace

Dans cette section, nous discutons des différents aspects de la sculpture d'espace propres à notre système. Nous abordons d'abord les structures internes d'images, nous voyons comment nous traitons la visibilité des voxels, nous expliquons comment les silhouettes sont sculptées et finalement, nous discutons de la comparaison de couleurs.

4.2.1 Structures internes d'images

Les images acquises par la caméra vidéo sont ajoutées à l'intérieur de notre système de reconstruction lorsqu'elles sont bien calibrées. Étant donné qu'il y a beaucoup d'images calibrées, il n'est pas possible de toutes les conserver, principalement pour deux raisons :

1. le manque d'espace mémoire ;
2. le temps de calcul, parce que beaucoup d'informations redondantes engorgeraient l'algorithme de reconstruction.

Afin de profiter le plus possible du grand nombre d'images entrant pour la sculpture de silhouettes tout en s'assurant d'avoir des images bien distribuées autour de l'objet pour

la comparaison de couleurs, nous avons mis en place une structure dynamique d'images et une structure statique d'images (Figure 4.6).

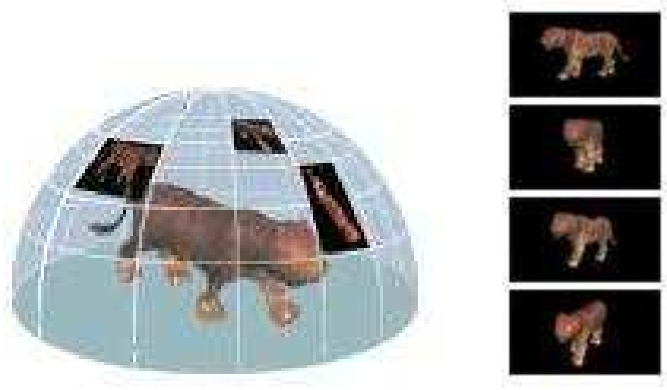

FIG. 4.6 – Gauche : structure statique d'images distribuées autour de l'objet. Droite : structure dynamique d'images traitées en FIFO.

Pour la structure statique, nous subdivisons l'hémisphère entourant l'objet en $\theta \times \phi$ régions (généralement 3×32) [GGSC96]. Nous conservons pour chaque région l'image calibrée dont la direction de vue se rapproche le plus du centre de sa région et orientée le plus vers l'objet. Cet ensemble d'images permet de raffiner le modèle 3D avec la comparaison de couleurs en tout temps. Il est intéressant de noter qu'appliquer l'algorithme uniquement sur ces images correspond à un algorithme de sculpture d'espace standard automatisé (rotation de l'objet ou bras robotisé).

Les images n'entrant pas dans la structure statique sont redirigées vers la structure dynamique (FIFO) à l'intérieur de laquelle les images sont traitées de la plus récente à la plus ancienne. Comme ces images ne restent pas nécessairement très longtemps en mémoire, elles servent principalement à la sculpture de silhouettes (Section 4.2.3). Cette structure contribue beaucoup à l'aspect interactif du système puisqu'elle permet de traiter rapidement beaucoup de silhouettes venant d'une même région, ce qui donne un retour visuel rapide à l'usager et le guide dans sa prise d'images. Cette structure contient en général 8 images.

Lors du processus d'acquisition d'images avec retour visuel, nous voulons donner la priorité aux régions privilégiées par l'usager (FIFO), mais voulons tout même avoir suffisamment de points de vue répartis autour de l'objet pour effectuer une comparaison de couleurs valide selon divers points de vue. Pour ce faire, nous traitons d'abord la structure dynamique et nous ajoutons un sous-ensemble aléatoire des images statiques

(entre 6 et 8 images). Nous avons remarqué que cela constituait un bon compromis.

4.2.2 Visibilité

Afin d'accumuler les couleurs des voxels valides et visibles dans les images courantes, nous projetons les voxels dans chacune de celles-ci sous forme de cubes OpenGL avec *backface culling*. Chaque voxel valide est coloré avec un ID unique encodé sur 32 bits dans une couleur conservée dans une table de hachage. Pour chaque image rendue avec ces cubes colorés, nous identifions chaque voxel visible par son ID et lui assignons les couleurs des pixels couverts dans l'image correspondante. Les IDs détectés ne servent en réalité qu'à déterminer quels voxels sont visibles dans une image. Lorsqu'un voxel est détecté visible, ses 8 coins sont projetés dans l'image et le rectangle enveloppant ces coins projetés forme la fenêtre de pixels couverte par ce voxel dans l'image (Figure 4.7). Toutes les couleurs des pixels à l'intérieur de cette fenêtre sont considérées lors de la comparaison de couleurs et aussi lors de la sculpture de silhouettes. Cette approche a

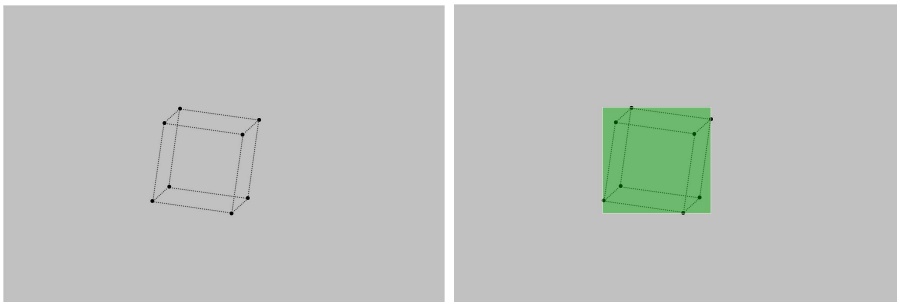

FIG. 4.7 – Pixels couverts par un voxel dans notre algorithme de visibilité. Gauche : projection des 8 coins du voxel dans l'image. Droite : rectangle enveloppant les 8 coins projetés.

pour effet de considérer potentiellement plus de pixels que ceux réellement couverts par le voxel. Par contre, étant donné que la comparaison de couleur que nous appliquons (Section 4.2.4) cherche à trouver la couleur la plus similaire à travers plusieurs listes de couleurs, il est plus conservateur de considérer plus de couleurs que moins. De la même manière, il est plus conservateur pour la sculpture de silhouettes de considérer plus de pixels. Effectivement, cela augmente la probabilité qu'un voxel se projetant près d'une silhouette dans une image touche à l'intérieur de l'objet et que par conséquent il ne soit pas rejeté. Aussi, considérer une plus grande couverture des voxels dans les images

permet de réduire les problèmes potentiels reliés à une calibration non-parfaite des images. Cette approche entraîne une enveloppe plus grande de l'objet puisque moins de voxels près de la silhouette sont éliminés. Heureusement, deux aspects de notre système permettent de contrer ce problème. D'abord, grâce à la grande quantité d'images sculptant les silhouettes, nous pouvons tout de même bien sculpter le modèle 3D. Ensuite la hiérarchie permet de subdiviser un voxel trop gros, et à son niveau plus détaillé, certaines parties à l'extérieur de l'objet pourront alors être éliminées.

Un problème qui peut se produire en traitant la visibilité avec OpenGL, et par conséquent avec un *z-buffer*, est qu'un voxel couvant moins d'un pixel ne soit pas considéré comme visible alors qu'il l'est en réalité. Ce problème n'est cependant pas critique dans notre cas puisque nous choisissons la taille d'un voxel dans la hiérarchie de telle sorte qu'il couvre toujours plus de 4 pixels.

Après avoir appliqué ce traitement à toutes les images disponibles, chaque voxel possède n listes de couleurs (si le voxel couvre plus qu'un pixel) associées aux n images dans lesquelles il est visible. Ces listes permettent d'appliquer la comparaison de couleurs appropriée telle que décrite dans la Section 4.2.4. Étant donné qu'à chaque itération des voxels peuvent être éliminés par sculpture, il est plus simple de projeter tous les voxels valides à chaque fois plutôt que de conserver l'information sur quels voxels nécessitent d'être mis à jour. Le processus pourrait cependant être accéléré en exploitant davantage la cohérence 3D de la structure hiérarchique.

4.2.3 Sculpture de silhouettes

Afin de traiter efficacement l'élimination des voxels projetant à l'extérieur des pixels couverts par l'objet dans les images courantes, il est nécessaire d'effectuer un traitement spécial sur chacune des nouvelles images ajoutées aux structures internes d'images du système. Pour chaque image, l'arrière-plan de l'objet est segmenté de manière à obtenir une image binaire (Section 4.4.2, Figure 4.10) qui est conservée avec son image correspondante. Lors de l'identification des voxels visibles dans une image, tous les voxels ne couvrant aucun pixel associé à l'objet dans l'image segmentée sont automatiquement rejetés. Ceci permet d'éliminer rapidement des voxels à chaque image traitée lors du processus de visibilité des voxels donnant un retour visuel rapide à l'usager lorsqu'il ajoute de nouvelles images. Ainsi, l'usager est guidé en constatant si l'ajout de nouvelles

images d'une région de l'objet permet de sculpter davantage l'objet avec les silhouettes.

Comme seuls les voxels visibles dans une image peuvent être éliminés, plusieurs itération d'identification des voxels visibles doivent être effectuées sur une même image pour éliminer tous les voxels à l'extérieur des silhouettes. Certaines techniques, comme le *ray marching* [Gla89], pourraient être appliquées pour traiter efficacement les silhouettes avec une seule itération de visibilité. Étant donné que le traitement de la visibilité avec OpenGL est très rapide et que les images dynamiques sont utilisées plusieurs fois durant leur existence dans notre système, nous n'avons pas insisté sur la pertinence d'optimiser cette partie de l'algorithme.

Ce traitement contribue énormément à l'interactivité du système par rapport à la comparaison de couleurs. Effectivement, nous traitons environ 4 images par seconde (visibilité et sculpture de silhouettes) pour une résolution de $128 \times 128 \times 128$ voxels. L'usager voit ainsi rapidement et continuellement la progression de la reconstruction. La comparaison de couleurs demeure un processus moins interactif puisqu'il est nécessaire d'accumuler les couleurs de plusieurs images avant de pouvoir effectuer la comparaison. En supposant que 20 images sont traitées, l'accumulation de couleurs prend environ 5 secondes. Il peut nécessiter plusieurs itérations de comparaisons de couleurs avant que l'algorithme converge vers une solution, ce qui prend par conséquent un temps de calcul non négligeable. Ce traitement, décrit dans la prochaine section, est donc en général appliqué sur demande de l'usager pour améliorer la solution obtenue avec la sculpture de silhouettes.

4.2.4 Comparaison de couleurs

La sculpture de silhouettes permet d'éliminer des parties extérieures à l'objet. Par contre, il est impossible avec cette technique de creuser dans certaines sections concaves de l'objet. Dans la mesure où l'objet contient des surfaces texturées, la comparaison de couleurs permet de creuser dans ces parties de l'objet où la sculpture de silhouettes ne peut rien faire. Il est à noter que le coeur de l'algorithme de sculpture d'espace [KS00] réside dans la comparaison de couleurs. Plus la comparaison est bien adaptée au type de surfaces de l'objet à reconstruire, plus il sera possible de retirer de l'information à partir des images permettant une meilleure reconstruction.

En supposant une illumination fixe, la couleur d'un point d'une surface diffuse (lam-

bertienne) reste invariante au point de vue. En exploitant cette propriété, il est possible de rejeter un voxel en observant la variance de la couleur des pixels des images dans lesquelles il est visible [SD99, KS00]. Comme il existe un certain bruit dans les images, que les surfaces sont rarement parfaitement diffuses, qu'un pixel n'est pas toujours entièrement couvert par une surface de même normale ou couleur, etc., il est nécessaire d'utiliser un seuil en dessous duquel un voxel est rejeté. Un seuil trop sévère peut engendrer une détérioration de l'objet due à de faux rejets. Par contre, un seuil trop conservateur empêche potentiellement de creuser des régions de l'objet.

Une autre supposition faite dans la comparaison standard de couleurs avec variance est qu'un voxel projette dans environ 1 pixel dans toutes les images. Malheureusement, les images peuvent être prises à différentes distances entraînant différentes tailles de voxels dans les images. Aussi, l'erreur de calibration peut induire que la bonne couleur se trouve en réalité dans un pixel voisin de celui dans lequel le voxel est identifié. Kutulakos [Kut00] propose le test de *r-consistence* comme solution à ces deux problèmes. Ce test cherche une couleur "similaire" à l'intérieur d'un rayon de dimension r à travers toutes les images visibles d'un voxel. Si une telle couleur est trouvée dans les images et que la variance des couleurs "similaires" est inférieure au seuil défini, le voxel ne peut être rejeté. Cette extension entraîne un volume englobant plus grand de l'objet. Par contre, l'objet a moins de risques d'être faussement sculpté par l'algorithme. Nous avons remarqué que cette approche menait à de meilleurs résultats dans notre système. Nous considérons tous les pixels couverts par le voxel pour la comparaison de couleurs si le voxel couvre plus de 16 pixels dans une image. Sinon nous considérons une fenêtre de 4×4 pixels entourant la projection du centre du voxel. La couleur la plus similaire à travers toutes les images dans lesquelles un voxel est visible lui sera assignée pour l'affichage. Ce choix de couleur donne des résultats visuels plus plaisants. L'Annexe C donne une description de cet algorithme. Nous avons également décidé d'utiliser cette approche avec la sculpture des silhouettes, pour palier aux imprécisions de calibration.

À chaque fois que le piédestal est déplacé, l'illumination sur l'objet est changée. Tourner l'objet équivaut à tourner la lumière. Étant donné que la comparaison de couleurs suppose une illumination fixe, il est très important de considérer ce changement d'illumination. Comme plusieurs images sont prises pour chacune des positions du piédestal, nous effectuons la comparaison de couleurs (Section 4.2.4) sur chacun des

groupes d'images possédant la même illumination. Un voxel ne sera rejeté que s'il échoue un de ces tests.

4.3 Niveaux de détail

Normalement, la sculpture d'espace produit de meilleurs modèles 3D lorsque tous les voxels visibles se projettent dans environ un pixel. Ce n'est malheureusement pas souvent le cas puisqu'il est possible d'acquérir des images à différentes distances de l'objet et, de plus, il peut parfois être intéressant d'extraire différents niveaux de détail de cet objet. Nous utilisons donc une grille régulière hiérarchique avec 8 enfants (octree) pour permettre de s'adapter aux bons niveaux de détail en fonction des images acquises par l'usager.

À chaque niveau, le maximum de voxels est rejeté avant de subdiviser pour atteindre le prochain niveau (plus détaillé). Il est ainsi possible de sculpter de grosses portions de l'espace à des niveaux moins détaillés, ce qui permet d'accélérer l'algorithme. Seuls les voxels visibles dans au moins une image sont subdivisés, afin d'économiser de l'espace mémoire. Effectivement, seulement la surface de l'objet se trouve à être subdivisée. Comme la structure de voxels repose dans l'espace tridimensionnel, l'espace mémoire occupé par les voxels est de l'ordre $O(n^3)$. Mais puisque nous ne subdivisons que les voxels à la surface de l'objet, le nombre de voxels se réduit à $O(n^2)$. Grâce à cette économie, il est possible de produire des résolutions de modèles plus élevées par rapport à une pleine résolution.

Lorsqu'une nouvelle image est insérée dans le système, elle est classifiée à un certain niveau de subdivision de manière à ce que les voxels projettent dans moins de 2 pixels en moyenne. Lors du traitement de la visibilité, les voxels sont projetés au niveau de subdivision approprié de l'image traitée. Ceci permet de traiter les voxels à différents niveaux de subdivision en tenant compte des images contenant de l'information pertinente pour chacun de ces niveaux.

Il est donc clair que l'utilisation d'une telle structure nous apporte beaucoup d'avantages : économie d'espace mémoire, accélération de la sculpture d'espace et traitement approprié de l'information.

4.4 Traitement des images

Afin de calibrer efficacement la caméra, sculpter les silhouettes, identifier les marqueurs sur le piédestal, etc., les images entrantes doivent être justement nettoyées. Il faut également considérer dans notre caméra (FIREWIRE) un certain bruit dans les images dû à une légère compression JPEG et à une illumination contrastée engendrée par l'utilisation d'un projecteur. Pour ces raisons, nous avons développé un outil extrêmement utile, l'ellipsoïde RGB.

Tous les traitements décrits dans cette section sont effectués dans la phase d'initialisation du système. Cette phase d'analyse n'est nécessaire que si les conditions d'illumination ont changé, que la calibration des couleurs de la caméra a été modifiée ou encore que la couleur de l'environnement de reconstruction a changé. Dans le cas contraire, c'est-à-dire si les trois dernières conditions sont respectées, comme les ellipsoïdes sont sauvegardés, le système est automatiquement initialisé avec la configuration de la dernière utilisation.

4.4.1 Ellipsoïde dans l'espace RGB

Étant donné le besoin d'identifier des couleurs dans les images pour différentes tâches, nous avons analysé la réponse de couleur de la caméra pour différentes couleurs projetées par le projecteur. Pour chaque couleur observée, nous avons récolté les pixels de la caméra correspondant aux zones de l'environnement de reconstruction illuminées par des fenêtres de 4×4 pixels du projecteur. Nous avons observé que cette distribution de couleurs ressemble à une distribution gaussienne et par conséquent, peut s'approximer par un ellipsoïde dans l'espace RGB.

Nous avons privilégié l'espace de couleur RGB aux autres espaces de couleurs, comme les espaces CIE XYZ, HSV ou des espaces plus perceptuels comme les espaces CIE LAB ou CIE Luv. Nos expérimentations avec ces différents espaces nous ont permis de conclure que l'espace RGB apportait des résultats similaires aux autres espaces. De plus, l'espace RGB permet un control plus simple et intuitif pour l'usager et est plus rapide à traiter puisqu'il n'y a aucune transformation d'espace à effectuer (les données acquises sont en RGB). Malgré tout, la couleur étant au coeur de la sculpture d'espace, des expérimentations dans les espaces de couleurs plus perceptuels devraient être investiguées.

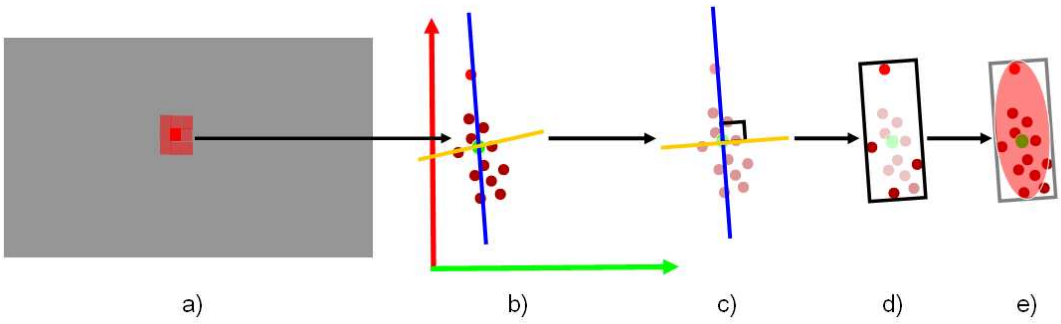

FIG. 4.8 – Illustration de la construction d'un ellipsoïde de couleurs. a) Pixels détectés dans une image. b) PCA sur la distribution de points de couleurs dans l'espace RGB, en 2D ici pour l'illustration. Le point vert représente la couleur moyenne de la distribution sur laquelle repose les principaux axes retrouvés avec le PCA. c) Rendre les axes perpendiculaires. d) Boîte englobante rectangulaire construite à l'aide des points extrêmes suivants les axes. e) Ellipsoïde à l'intérieur de la boîte englobante. L'ellipsoïde est une approximation, par conséquent, certains points de la distribution peuvent être à l'extérieur de celui-ci.

Pour construire ces ellipsoïdes, nous appliquons d'abord une analyse des composantes principales (PCA) [PFTV92] sur l'ensemble des couleurs des pixels appartenant à un point de couleur projeté. Cette analyse nous permet d'identifier les orientations principales de l'ensemble de couleurs à partir desquelles il est possible d'identifier la boîte englobante orientée de ces points. Cependant, comme les orientations retrouvées avec le PCA ne sont pas nécessairement perpendiculaires, il faut appliquer une petite manipulation mathématique pour s'assurer d'avoir une boîte engloblante rectangulaire. Pour ce faire, nous obtenons dans un premier temps un vecteur perpendiculaire aux deux principaux axes avec le produit vectoriel de ces derniers. Finalement, le produit vectoriel du nouveau vecteur avec le vecteur représentant la principale orientation de la distribution de points donne l'autre axe formant ainsi la base perpendiculaire recherchée. La boîte englobante est donc construite avec ces trois axes et ses dimensions sont déterminées par les points extrêmes de la distribution de telle sorte que tous les points soient compris à l'intérieur de la boîte.

Nous avons remarqué que cette boîte contenait peu de points aux extrémités, s'apparentant à une distribution gaussienne. De plus, ces boîtes avaient tendance à s'intersecter dans les régions près de l'origine (i.e du noir). C'est pourquoi nous utilisons à la place

le plus gros ellipsoïde orienté à l'intérieur de la boîte englobante. Il arrive ainsi moins de fausses identifications puisque la distribution de couleurs est mieux approximée dans l'espace RGB (Figure 4.8). Par contre, il est à noter que l'ellipsoïde est une approximation de la distribution de points et par conséquent il se peut que certains points extrêmes ne soient pas inclus dans l'ellipsoïde.

Un autre problème rencontré lors de cette analyse est relié aux couleurs saturées. Malheureusement, la caméra possède une plage de couleurs limitée, ce qui entraîne une saturation sur un ou plusieurs axes de couleur (R,G,B) lorsque l'intensité de la lumière est trop intense. Ceci se traduit par des changements d'orientation sur les différentes faces du cube RGB. En réalité, au maximum deux changements d'orientation peuvent se produire dus à la saturation, aux extrémités des axes R, G ou B. Cette saturation produit une boîte englobante trop grande et mal orientée si elle n'est pas bien traitée. Lors de la construction de l'ellipsoïde, nous traitons séparément les deux groupes de pixels saturés, s'ils existent. Dans un premier temps la boîte englobante des pixels non saturés (RGB $\in$ [0,254]) est construite. Ensuite, nous construisons la deuxième boîte englobante sur le premier groupe de pixels saturés et calculons la rotation nécessaire pour l'aligner avec la précédente. Le même processus est appliqué sur le deuxième groupe de pixels saturés si nécessaire. Finalement, l'ellipsoïde englobante tous les points (comprenant les pixels saturés transformés) est construit. La Figure 4.9 illustre en 2D ce traitement sur les pixels saturés.

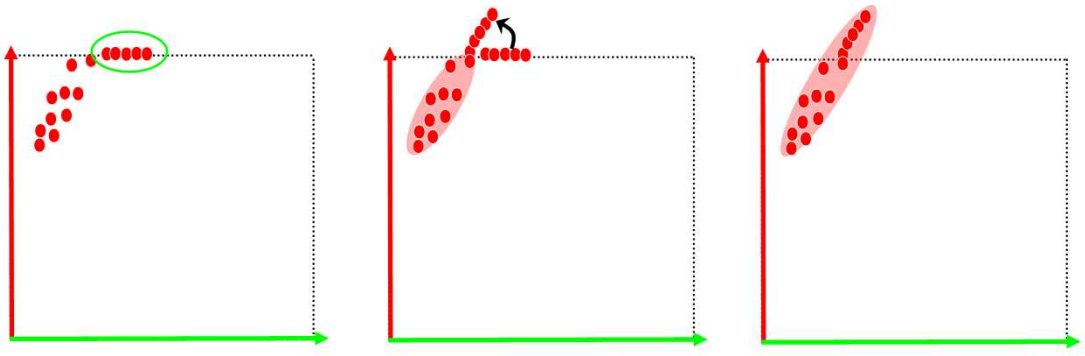

FIG. 4.9 – Traitement sur les pixels saturés dans la construction des ellipsoïdes. Gauche : identification des couleurs saturées. Centre : rotation sur le groupe de couleurs saturées pour l'aligner avec le groupe de couleurs non saturées. Droite : nouvel ellipsoïde englobant toutes les couleurs.

Une fois ce dernier ellipsoïde construit, l'identification d'une couleur appartenant à celui-ci est directe. Nous connaissons la transformation rigide (encodée dans une matrice) permettant de transformer l'ellipsoïde orienté en une sphère centrée à l'origine et de rayon unitaire. L'identification résulte donc dans le test d'intériorité de la couleur transformée dans l'espace sphère en suivant Formule 4.1.

$$(\frac{R}{255})^2 + (\frac{G}{255})^2 + (\frac{B}{255})^2 \leq 1. \tag{4.1}$$

Dans les prochaines sections, nous décrivons de quelle manière nous utilisons cet outil dans notre système.

4.4.2 Segmentation de l'objet

Afin de pouvoir procéder à la sculpture des silhouettes (Section 4.2.3), il faut déterminer quelles parties de l'image ne sont pas couvertes par l'objet à reconstruire. Comme la caméra se déplace continuellement, il n'est pas possible d'effectuer une soustraction des images avec et sans l'objet. Par contre, nous pouvons exploiter la connaissance des couleurs des panneaux de l'environnement de reconstruction pour effectuer une segmentation efficace.

En prétraitement, nous construisons un ellipsoïde en recueillant les pixels correspondant à l'arrière-plan illuminé par le projecteur (Figure 4.12 centre). Comme la caméra est fixe et calibrée lors de ce processus, nous connaissons la position des pixels correspondant à l'arrière-plan illuminé par le projecteur. Par la suite, nous utilisons cet ellipsoïde pour segmenter toutes les images entrantes dans le système. La Figure 4.10 montre une image de la caméra accompagnée de son image segmentée.

La limitation de cette approche est que l'objet ne doit pas contenir la même couleur que l'arrière-plan. Si ce n'est pas le cas, il sera faussement segmenté et par conséquent des voxels valides seront éliminés lors de la sculpture des silhouettes. En général, comme la lumière éclairant l'objet est blanche, les spécularités de l'objet peuvent aussi apparaître blanches. Pour cette raison, nous utilisons des cartons de couleur (en général vert ou bleu, dépendant des couleurs de l'objet) comme arrière-plan, ce qui mène à une meilleure segmentation.

Comme nous voulons toujours être conservateur pour ne jamais éliminer des voxels valides, nous devons nous assurer que l'objet est toujours réellement compris à l'intérieur

de la zone identifiée par la segmentation ellipsoïdale. Pour ce faire, nous appliquons un filtre de dilatation de un pixel sur l'image segmentée pour agrandir la zone couverte par l'objet et aussi pour éliminer des pixels isolés de l'objet faussement segmentés.

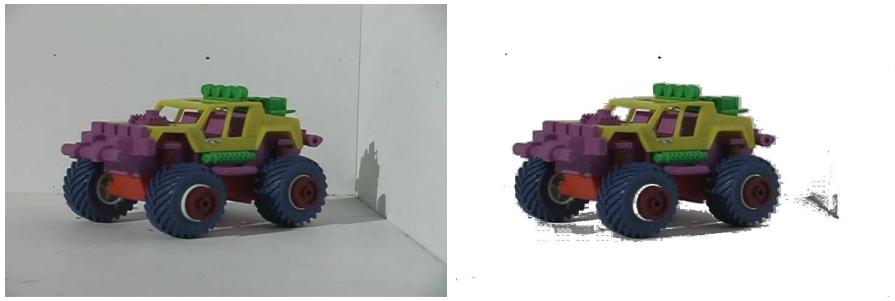

FIG. 4.10 – Gauche : image originale. Droite : image segmentée sans les pixels appartenant à l'ellipsoïde des couleurs de l'arrière-plan.

4.4.3 Identification des points de calibration

Tel que décrit dans la Section 4.1.2, les positions des points de calibration projetés par le projecteur doivent être retrouvées efficacement et avec précision dans les images entrantes. Lors de l'initialisation, nous construisons un premier ellipsoïde à partir des pixels d'une image de la scène sans illumination. Cet ellipsoïde permet d'éliminer l'illumination ambiante de la salle dans les images, qui est en général assez faible mais qui peut tout de même varier selon l'humeur des membres des laboratoires environnants. Ensuite, nous projetons plusieurs points de couleur choisie pour des points de calibration sur les différentes faces de l'environnement de reconstruction. À l'aide de l'ellipsoïde d'illumination ambiante précédemment construit, tous les pixels de l'image non illuminés par le projecteur sont éliminés de telle sorte que seuls les pixels illuminés sont récoltés (Figure 4.11 gauche). En appliquant ce traitement sur toutes les couleurs utilisées pour les points de calibration de la caméra, nous construisons les ellipsoïdes permettant une robuste et rapide calibration automatique (Figure 4.11 droite).

Nous avons essayé différentes couleurs pour les points de calibration et nous avons constaté que plusieurs des ellipsoïdes correspondants s'intersectaient dans l'espace RGB dû à la réponse de la caméra vidéo utilisée. Une intersection d'ellipsoïdes mène à de mauvaises identifications des pixels, entraînant une mauvaise calibration. Pour éviter ce problème, nous n'utilisons que les couleurs rouge, vert, bleu et gris, lesquelles ne

s'intersectent pas.

Deux modes de calibration existent dans le système (Section 4.1.2), un sécuritaire à l'intérieur duquel l'objet n'est pas éclairé facilitant la détection des points de calibration, et un temporel où la dernière position de chaque point sert comme centre d'une fenêtre de recherche pour la nouvelle position.

Dans le mode sécuritaire, l'ellipsoïde d'illumination ambiante est d'abord utilisé pour éliminer le plus grand nombre de pixels inutiles. Ensuite, à l'aide des ellipsoïdes des points de calibration rouge, vert et bleu, nous identifions tous les pixels correspondant respectivement à chacun de ces groupes. Nous savons que pour chacun de ces groupes il existe deux points projetés, un au-dessus de l'autre. En supposant que le *up vector* de la caméra reste toujours vers le haut, les deux points sont facilement séparés par un algorithme de groupement 2-moyenne et ainsi uniquement identifiés. Ceci nous donne 6 points de calibration, mais en réalité il nous en faut au moins 8. C'est pourquoi nous ajoutons des points gris. Ces points sont situés entre des points de couleur (Figure 4.2) connus. Il suffit donc de rechercher les pixels appartenant à l'ellipsoïde de gris à l'intérieur d'un rectangle en 2D formé par deux points de couleur entourant un point gris. Nous cherchons donc 6 points de calibration supplémentaires avec ces points gris, pour un total de 12 points, ce qui est amplement suffisant. Même si certains de ces points sont mal identifiés, il est possible de les rejeter et de calibrer tout de même la caméra.

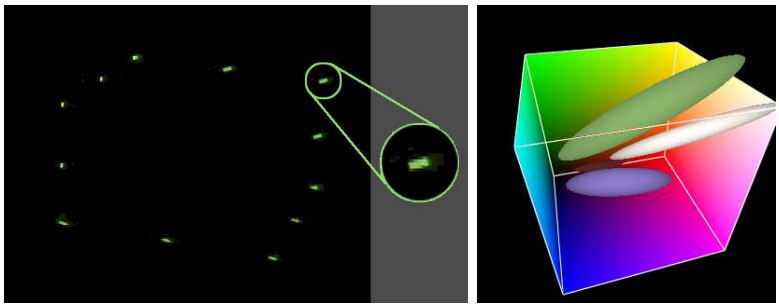

FIG. 4.11 – Gauche : une image prise en prétraitement pour analyser la distribution des points de calibration verts. Droite : les 4 ellipsoïdes dans l'espace RGB des différents points de couleur de calibration.

Le mode temporel est beaucoup plus rapide puisqu'il suffit d'utiliser l'ellipsoïde de la couleur recherchée à l'intérieur d'une fenêtre de 40×40 pixels. Cependant, une

illumination indirecte causée par la réflexion de la lumière projetée sur l'objet nuit parfois à l'identification des points gris. En effet, cette illumination indirecte est souvent grise sombre et interfère avec la partie inférieure de l'ellipsoïde de gris. Pour remédier à ce problème, nous précalculons un ellipsoïde d'illumination indirecte de la même façon que celui de l'arrière-plan (Section 4.4.2) sauf que nous récoltons les pixels à l'extérieur de la zone illuminée plutôt qu'à l'intérieur (Figure 4.12 gauche). Nous utilisons ensuite cet ellipsoïde sur les images acquises dans le mode temporel avant d'utiliser les autres ellipsoïdes.

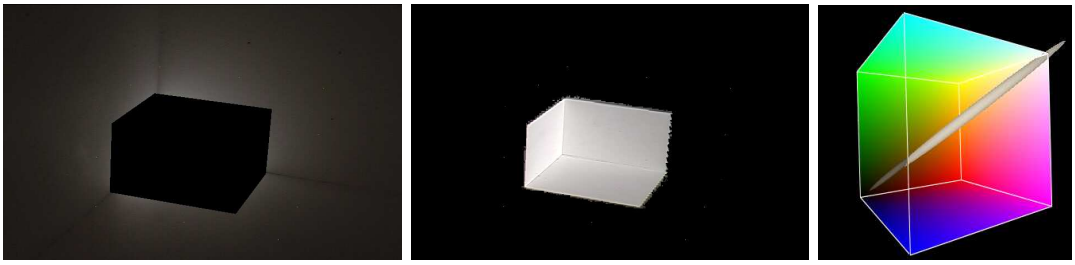

FIG. 4.12 – Gauche : image servant à construire l'ellipsoïde d'inter-réflexion. Les pixels dans la région éclairée sont masqués. Centre : l'arrière-plan illuminé par le projecteur. Droite : les deux ellipsoïdes correspondants.

4.5 Affichage

Afin de fournir un retour visuel à l'usager lors de la reconstruction interactive, l'affichage du modèle 3D en cours de reconstruction se doit d'être efficace et approprié. Dans un premier temps, l'affichage ne doit pas trop ralentir les différentes opérations en cours. Pour ce faire, nous projetons chaque voxel sous forme de cube OpenGL, lequel est constitué en un évantail de triangles (*triangle fan*). Après quelques tests, cette implémentation s'est avérée la plus performante. Dans un deuxième temps, pour guider l'usager sur l'effet produit par les images récemment acquises, le modèle 3D est affiché du point de vue de la dernière image calibrée lors de la reconstruction.

La couleur des voxels peut être déterminée de deux façons. Lors de la comparaison de couleurs, la couleur la plus "plausible" selon toutes les couleurs récoltées est conservée pour le voxel. Cette couleur est assignée à tout le voxel. En général, ce choix de couleur est très satisfaisant. Il peut être intéressant cependant de capter les effets dépendants du point de vue, comme les spécularités, tout en ayant un affichage efficace. Pour ce faire,

nous utilisons une combinaison des trois images les plus proches du point de vue virtuel désiré pondéré par la distance angulaire. Dans notre implémentation, un ensemble de 42,000 voxels sont projetés en OpenGL à raison de 18 fps (*frames par seconde*) alors qu'avec les projections de textures, cet affichage descend à environ 8 fps. La Figure 4.13 montre les différents modes d'affichage dans notre système.

FIG. 4.13 – Affichage du modèle 3D dans notre application. Gauche : une couleur par voxel. Droite : reprojection de textures pour mieux capter les spécularités. Il s'agit d'une interpolation des trois images les plus proches du points de vue virtuel désiré.

Étant donné que nous avons une structure hiérarchique, il est possible de visualiser le modèle aux différents niveaux. Par défaut, les voxels les plus raffinés sont affichés. Cependant, l'usager peut spécifier le niveau désiré d'affichage. Cette option est pratique lorsque le modèle 3D atteint des grandes résolutions et que par conséquent l'affichage

ralentit. L'usager peut choisir une résolution moins grande pour manipuler l'objet rapidement pour ensuite revenir à la résolution désirée. Il aurait été possible d'automatiser ce processus en assurant un temps de rendu minimal, en dessous duquel des niveaux supérieurs sont choisis [RHHL02]. Cette idée n'a pas été implémentée faute de temps. L'information sur les couleurs obtenue aux niveaux plus détaillés peut être propagée vers les niveaux moins détaillés, procurant une meilleure cohérence entre ceux-ci lors de la visualisation (Figure 4.14).

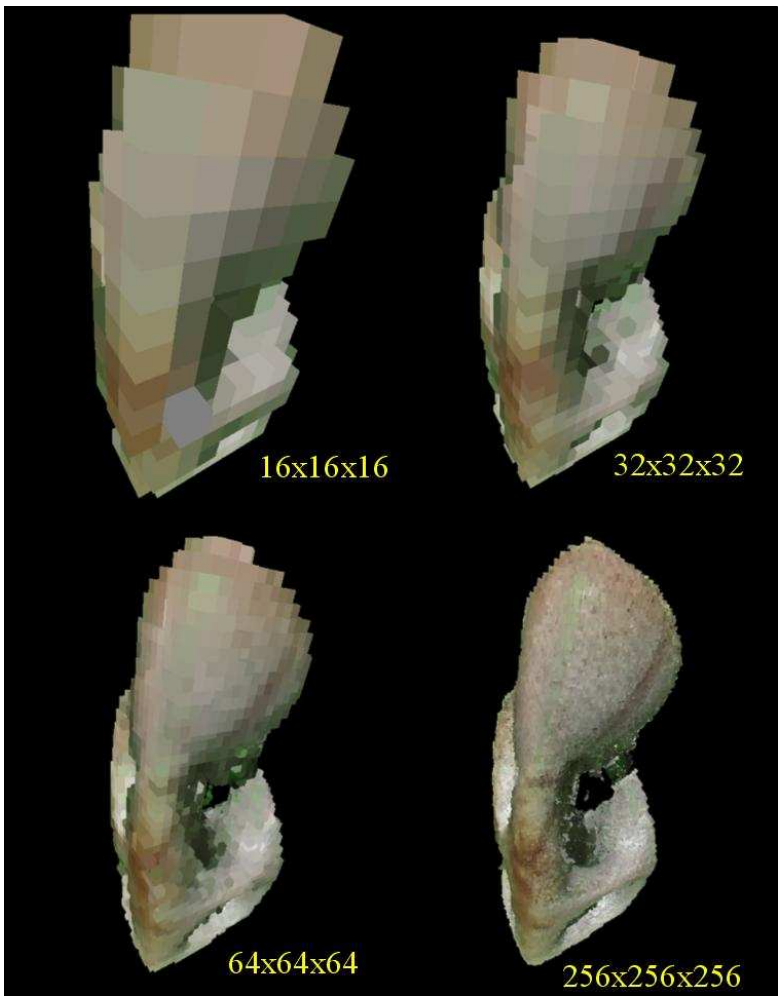

FIG. 4.14 – Affichage sous forme de voxels de différents niveaux de la hiérarchie d'un même modèle.

4.6 Exportation du modèle

Une fois la reconstruction satisfaisante, il est possible d'exporter le modèle sous forme de points augmentés d'une normale et d'une couleur. Nous utilisons ensuite l'application *QSplat* [RL00] pour afficher en temps réel un très grand nombre de points. Tous les résultats présentés au Chapitre 5 sont affichés dans ce système. Le modèle sauvegardé perd ses informations de notre structure hiérarchique puisque nous sauvegardons seulement des points à la résolution désirée par l'usager.

Les normales à la surface sont calculées à l'aide d'un PCA. La normale correspond à la normale du plan passant par l'ensemble de voxels valides voisins du voxel traité. Ce plan est calculé avec le PCA. Plus le rayon des voisins considérés autour des voxels est grand, plus les normales sont lissées. Il n'est pas évident de déterminer la bonne distance de recherche pour les voisins. De plus, il pourrait être intéressant de calculer des cônes de normales à partir des différents niveaux de subdivision afin d'accélerer l'affichage. La taille d'affichage des *splats* est deux fois la distance avec le voxel voisin au même niveau de la hiérarchie.

Ceci termine la description de notre système. Dans le prochain chapitre, nous discutons de nos résultats et nous montrons plusieurs modèles 3D reconstruits avec notre système.

Chapitre 5

Résultats

Toutes les expérimentations présentées dans ce chapitre ont été effectuées sur un Pentium IV Xeon biprocesseur 2.4 GHz avec 1 GB de mémoire et une carte graphique nVIDIA GeForce4 Ti 4200. Toutes les techniques développées dans notre système sont effectuées en temps interactif sur cette machine : acquisition, calibration et nettoyage des images, sculpture des silhouettes, identification des voxels, acquisition et comparaison de couleurs pour la sculpture par couleur, affichage des voxels colorés, etc.

L'acquisition de la vidéo numérique est effectuée à partir d'une caméra Panasonic PVGS-70 à une résolution de 720×480 pixels. Les images sont *reçues* par l'ordinateur sur une connexion *firewire* à un taux de transfert de 29.97 fps, avec une légère compression JPEG. Le projecteur DLP Compaq MP4800 émet 2100 lumens avec une résolution de 1024×768 pixels.

Nous avons construit deux environnements de reconstruction de tailles différentes pour permettre la reconstruction d'objets de grosseurs variées. Le premier mesure $81 \times 61 \times 63$ cm alors que le deuxième mesure $25.4 \times 25.4 \times 25.4$ cm. Ils sont peints avec de la peinture mate blanche. 25 petites croix sont tracées à des positions mesurées afin de faciliter la calibration manuelle du projecteur.

La calibration d'une image, incluant le temps de la nettoyer, identifier les points de calibration et segmenter l'objet de l'arrière-plan prend environ 0.17 seconde. En moyenne, nous obtenons une erreur sous pixel pour la reprojection des points de calibration 3D.

Le Tableau 5.1 montre un estimé moyen en secondes du temps requis pour exécuter la sculpture d'espace (silhouettes et couleur) en fonction de différentes résolutions de

grille de voxels. La deuxième colonne indique le temps moyen pour identifier les IDs des voxels et éliminer ceux qui se projettent à l'extérieur de la silhouette de l'objet. La dernière colonne indique le temps moyen pour appliquer la comparaison de couleurs sur les voxels une fois que les couleurs ont été recueillies durant l'étape de visibilité.

Résolution en espace voxel	sculpture des silhouettes	sculpture de couleur
32^3	0.06 sec.	1.3 sec.
64^3	0.15 sec.	1.9 sec.
128^3	0.25 sec.	3.2 sec.
256^3	0.50 sec.	4.6 sec.

TAB. 5.1 – Temps d'exécution.

Le Tableau 5.2 peint le portrait de la répartition du temps dans une séance typique de reconstruction. Les temps pour la reconstruction varient beaucoup selon la complexité du modèle à reconstruire et du niveau de détail désiré. Il est possible d'obtenir un modèle grossier de l'objet en moins de 5 minutes, ce qui permet à l'usager de développer une bonne intuition du temps nécessaire pour obtenir un modèle de la qualité désirée. Nous considérons que notre système permet de reconstruire rapidement un objet comparativement à une façon plus traditionnelle de procéder. Effectivement, la partie acquisition, calibration et nettoyage des images peut normalement prendre jusqu'à des heures alors que pour un temps presque équivalent, nous obtenons le modèle final. D'ailleurs, comme nous l'avons mentionné dans la Section 2.4, l'algorithme de sculpture d'espace de Kutulakos et Seitz [KS00] prend des temps un peu supérieurs aux nôtres soit environ 110 minutes pour un modèle 3D ayant une résolution de $256 \times 256 \times 256$. Il faut aussi souligner que ces temps ne comptent pas le temps d'acquisition et de calibration des images.

calibration du projecteur	2-3 mins
initialisation du système	1 min
reconstruction 3D interactive	10-30 mins
acquisition de détails de surface	10-20 mins
sculpture par couleur (optimisation finale)	10 mins et +

TAB. 5.2 – Répartition du temps lors d'une séance typique de reconstruction.

Dans la configuration actuelle et étant donnés les objets que nous reconstruisons, une résolution de 128^3 à 256^3 voxels hiérarchique donne des résultats satisfaisants alors

qu'une résolution de 512^3 voxels hiérarchique permet d'obtenir des détails de surface accrus. En général, les modèles 3D finaux contiennent quelques centaines de milliers de voxels (aux résolutions les plus subdivisées). Le nombre d'images acquises varient beaucoup selon la complexité des objets. En moyenne, environ 400 images sont traitées lors d'une session.

Nous n'avons pas établi de mesure d'erreur sur nos modèles reconstruits dans le cadre de cette maîtrise. Quoi qu'il serait fort intéressant de le faire, il s'agit en soi d'un domaine de recherche très complexe avec beaucoup de problèmes à explorer et solutionner.

Les prochaines sections de ce chapitre illustrent différents résultats obtenus avec notre système et discutent quelques détails spécifiques à chacun.

5.1 Petite statuette

Nous avons choisi cet objet pour illustrer la complexité de reconstruction des objets ayant plusieurs zones intérieures visibles que de certains points de vue (Figure 5.1). Étant donné les trous de cet objet et qu'il possède très peu de couleurs variées, c'est la sculpture par silhouettes qui procure les meilleurs résultats. La Figure 5.2 illustre le gain potentiel de pouvoir sculpter interactivement les silhouettes avec beaucoup d'images comparativement à une sculpture avec un ensemble d'images fixe bien distribuées autour de l'objet. Nous n'avons pas fait tourner l'objet lors de cette session puisque le but était

FIG. 5.1 – Images de la petite statuette.

de passer plus de temps à améliorer la qualité du modèle 3D interactivement. 800 images ont été acquises durant la session de reconstruction interactive qui a duré 40 minutes au

total. 20 images ont été conservées dans la structure statique d'images, ce qui correspond à un cadran de l'hémisphère. Effectivement, il y a $3 \times 32 = 96$ bins et comme l'objet n'a pas été tourné, seulement des images dans un cadran ont été prises, ce qui correspond à 24 bins. La sculpture par comparaison de couleurs a été appliquée hors ligne pendant environ 15 minutes. La résolution maximale atteinte du modèle 3D est de $256 \times 256 \times 256$.

5.2 Shiva

Cet objet illustre, comme la petite statuette, l'avantage de la sculpture interactive étant donné ses nombreuses silhouettes. Il n'a pas été facile de reconstruire cet objet parce qu'il possède des formes complexes, peu de variations de couleurs et beaucoup de détails de surface fins. La Figure 5.3 montre des images de cet objet. Nous avons pu ajouter plus de détail de surface de l'objet grâce à la calibration en gros plan avec des points de calibration projetés directement sur l'objet. La Figure 5.4 montre trois vues différentes du modèle 3D reconstruit à deux niveaux de détail différents et la Figure 5.5 montre en plus gros plan certaines régions du modèle, toujours à deux niveaux de détails. La structure statique a été remplie de 28 images alors que 437 images ont passé à travers la structure dynamique d'images. La période d'acquisition d'images et de reconstruction interactive a duré 35 minutes environ. La sculpture par comparaison de couleurs hors-ligne a été appliquée pendant 15 minutes. Afin d'obtenir plus de détails de surface et ainsi atteindre une résolution de $512 \times 512 \times 512$, 5 images en gros plan ont été acquises.

5.3 Chandelier avec Boo

Nous avons choisi cet exemple afin de montrer les avantages de la sculpture par comparaison de couleurs. Les silhouettes de cet objet permettent d'obtenir rapidement un modèle 3D intéressant. Par contre, l'intérieur du chandelier ne peut être sculpté uniquement par silhouettes. C'est pourquoi nous y avons placé des petites poupées russes colorées afin de laisser l'algorithme de sculpture par comparaison de couleurs creuser l'intérieur du chandelier. De plus, nous avons ajouté un petit fantôme nommé Boo sur le chandelier pour ajouter différents niveaux de détail au modèle 3D (Figure 5.6). En résumé, silhouettes pour le chandelier, comparaison de couleurs pour l'intérieur du chandelier et calibration en mode zoom pour Boo. La période d'acquisition d'images et de reconstruction a été assez rapide pour cet objet, environ 15 à 20 minutes puisque l'ajout de silhouettes après un certain temps n'apporte plus énormément d'informations à la reconstruction. D'ailleurs, on peut constater que beaucoup moins d'images ont été acquises au total que pour les deux précédents objets, soit 201 images. La structure statique a accumulé 17 images distribuées autour de l'objet. 8 images calibrées en mode zoom ont permis d'obtenir plus de détails sur le fantôme Boo. La résolution maximale du modèle 3D est de $256 \times 256 \times 256$. La Figure 5.7 montre 3 points de vue du modèle 3D qui pour illustrer que l'algorithme de comparaison de couleurs a bien réussi à creuser dans la concavité du chandelier. Des gros plans du fantôme et des poupées russes sont aussi présents dans cette figure.

5.4 Sac en peau de vache

Cet objet (Figure 5.8) est un autre exemple illustrant les avantages de la sculpture par comparaison de couleurs. Il y a des petits objets dans le sac qui ajoutent des détails plus fins au sac en peau de vache. Avec un objet de ce type, l'acquisition d'images et la reconstruction interactive ne prend pas beaucoup de temps comparativement à la sculpture par comparaison de couleurs (10-15 minutes vs. 30-60 minutes). Étant donnée la nature de cet objet, un nombre moins grand d'images est nécessaire pour sculpter les silhouettes. C'est pourquoi seulement 142 images ont été acquises durant la session de reconstruction, ce qui est beaucoup moins que la plupart des autres objets. 17 images ont été conservées dans la structure statique. 4 images calibrées en mode zoom ont servi à procurer plus de détails sur les objets dans le sac. La résolution maximale du modèle 3D est de $256 \times 256 \times 256$. Il est intéressant d'observer à la Figure 5.9 que l'algorithme a réussi à creuser profondément dans le sac. Par contre les parties dans l'ombre à l'intérieur du sac n'ont pas pu être sculptées. Ceci s'explique par le fait que ces régions sombres dans les images possèdent très peu de variation de couleurs et que par conséquent il est difficile d'éliminer un voxel de cette région en se basant sur la variance des couleurs. Il aurait fallu baisser plus bas le seuil de variance dans la comparaison de couleurs pour creuser davantage mais malheureusement, cela aurait eu pour effet de sculpter des régions valides du modèle 3D. Dans un tel cas, il serait intéressant de fournir à l'usager un outil interactif pour sélectionner une région 3D du modèle et n'appliquer la sculpture avec ses paramètres modifiés que sur cette région. Cet outil augmenterait encore plus les bénéfices d'une approche interactive de reconstruction.

5.5 Soldat chinois

Cet objet représente un soldat chinois (Figure 5.10). Il est relativement simple au niveau des formes et la sculpture par silhouettes est la technique qui procure les meilleurs résultats. Nous l'avons choisi comme exemple entre autres pour montrer un résultat de reconstruction avec le piédestal où l'objet est reconstruit selon des points de vue tout autour de l'objet. La session de reconstruction et d'acquisition d'images interactive est plus longue lorsque nous utilisons le piédestal puisque cela implique d'arrêter la reconstruction pour tourner l'objet, et cela plusieurs fois si on veut vraiment faire le tour de l'objet. Pour le soldat chinois, nous avons mis 40-45 minutes pour la reconstruction interactive en le tournant 3 fois. Par contre, comme l'objet possède peu de variations de couleurs, la sculpture par comparaison de couleurs n'a pris que 5-7 minutes. Au total, 294 images ont été acquises et 30 images ont été conservées dans la structure statique d'images. Étant donné que l'objet a été tourné 3 fois, il aurait été logique de conserver une soixantaine d'images (3 cadrans d'un hémisphère subdivisé en 3×32 régions). Ce qui peut expliquer le moins grand nombre d'images conservées est que l'usager ait pris beaucoup d'images dans la première position de l'objet et reconstruit ainsi une grande partie de l'objet. Ensuite, l'objet a été tourné pour terminer la reconstruction mais peu d'images ont été acquises dans le deux autres positions. La résolution maximale est de $256 \times 256 \times 256$. Les résultats de la reconstruction sont illustrés à la Figure 5.11.

Nous avons présenté dans ce chapitre des modèles qui illustrent les différentes capacités de notre système. D'autres exemples peuvent être visualisés sur la page web dédiée à ce projet [GP04]. Dans le prochain chapitre, nous allons conclure en discutant des résultats obtenus, en présentant nos observations sur les avantages et les inconvénients de notre système et en introduisant différentes possibilités de recherche dans le futur.

FIG. 5.2 – Gauche : reconstruction interactive. Droite : reconstruction avec images statiques distribuées dans les régions de l'hémisphère. On peut remarquer que les trous sont mieux définis à gauche car l'usager a pu choisir interactivement des points de vue selon lesquels ces trous étaient bien captés. Certaines régions correspondant à l'ombre générée par l'objet n'ont pas été éliminées, ce qui explique les amoncellements de voxels à l'extérieur du modèle 3D.

FIG. 5.3 – Images de Shiva.

FIG. 5.4 – Modèle 3D de Shiva. Gauche : $256 \times 256 \times 256$. Droite : $512 \times 512 \times 512$.

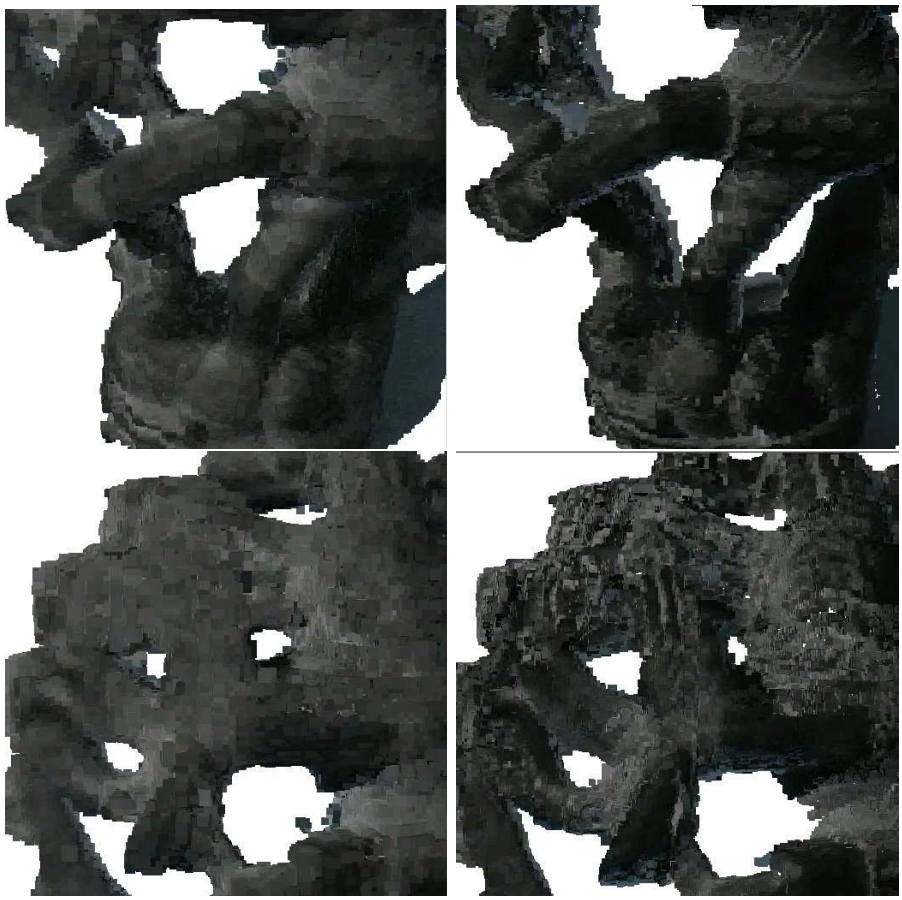

FIG. 5.5 – Modèle 3D de Shiva en gros plan. On peut y observer les détails de surface acquis en traitant des images en gros plan pour raffiner le modèle 3D en octree.

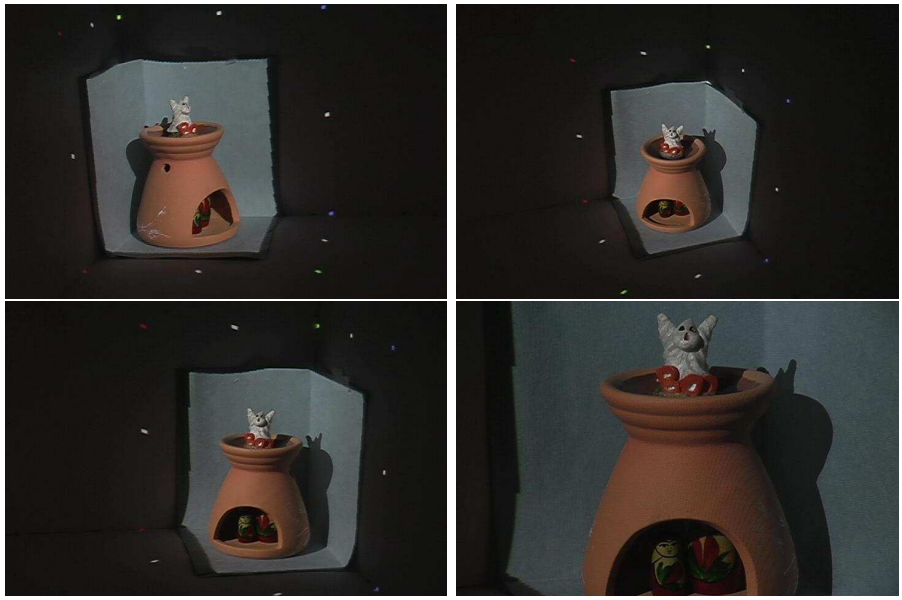

FIG. 5.6 – Images du chandelier avec Boo.

FIG. 5.7 – Modèle 3D du chandelier avec Boo.

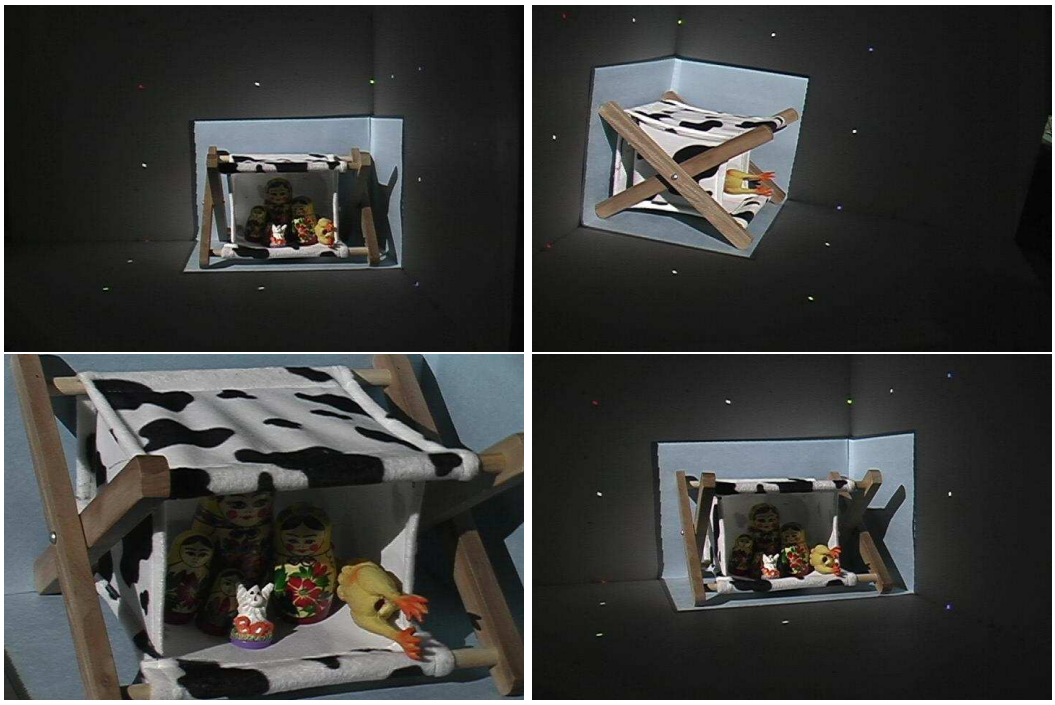

FIG. 5.8 – Images du sac en peau de vache.

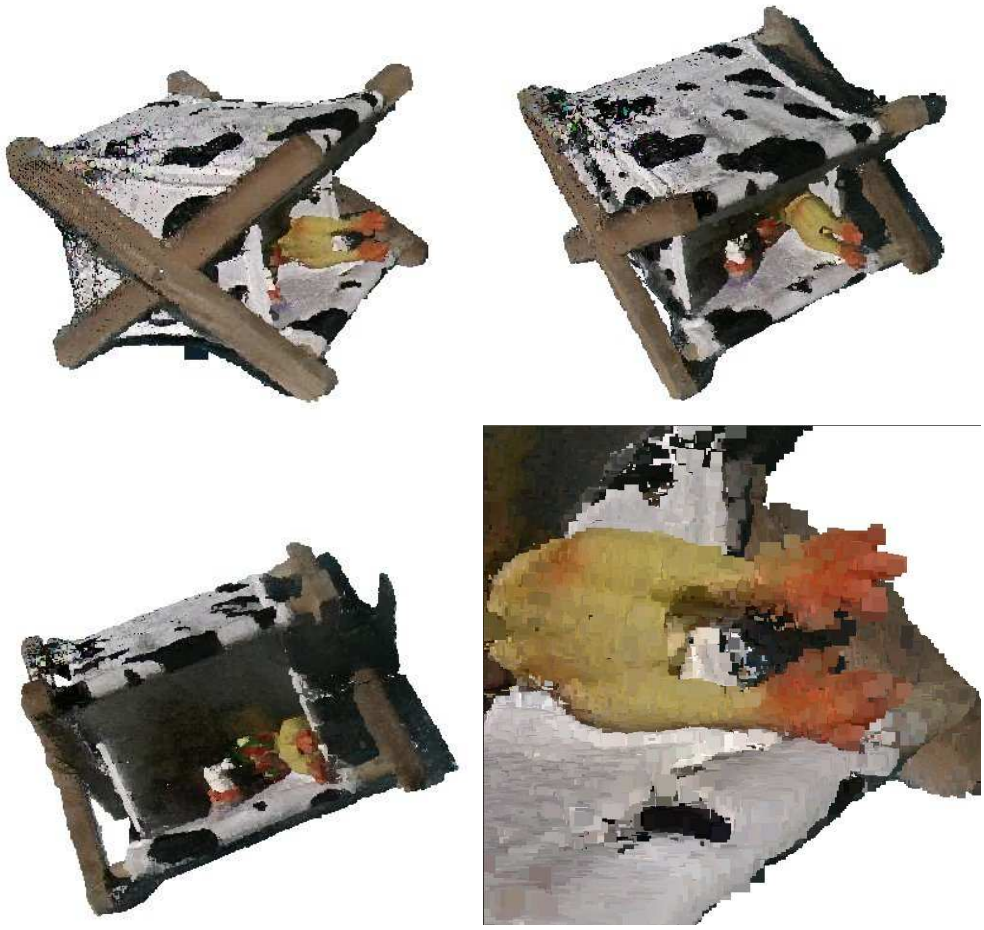

FIG. 5.9 – Modèle 3D du sac en peau de vache. Les zones intérieures éclairées sont bien sculptées par la comparaison de couleurs, mais les zones dans l'ombre le sont moins à cause du manque de variation de couleurs.

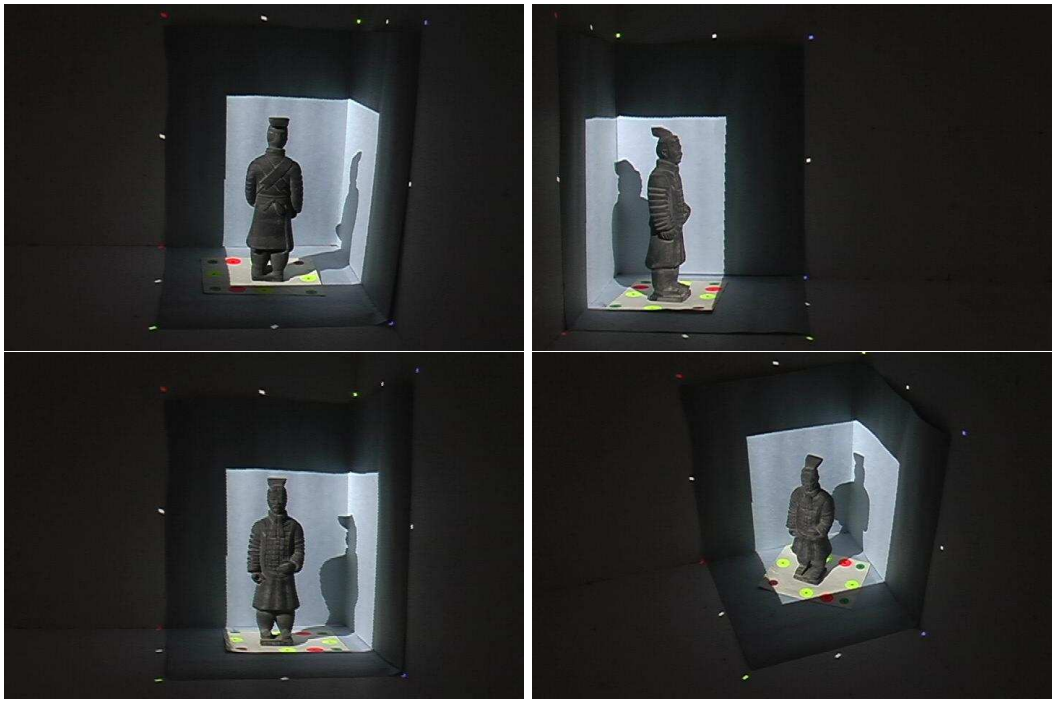

FIG. 5.10 – Images du soldat chinois. Le piédestal a permis de tourner l'objet à 3 reprises pour obtenir une reconstruction tout autour de l'objet.

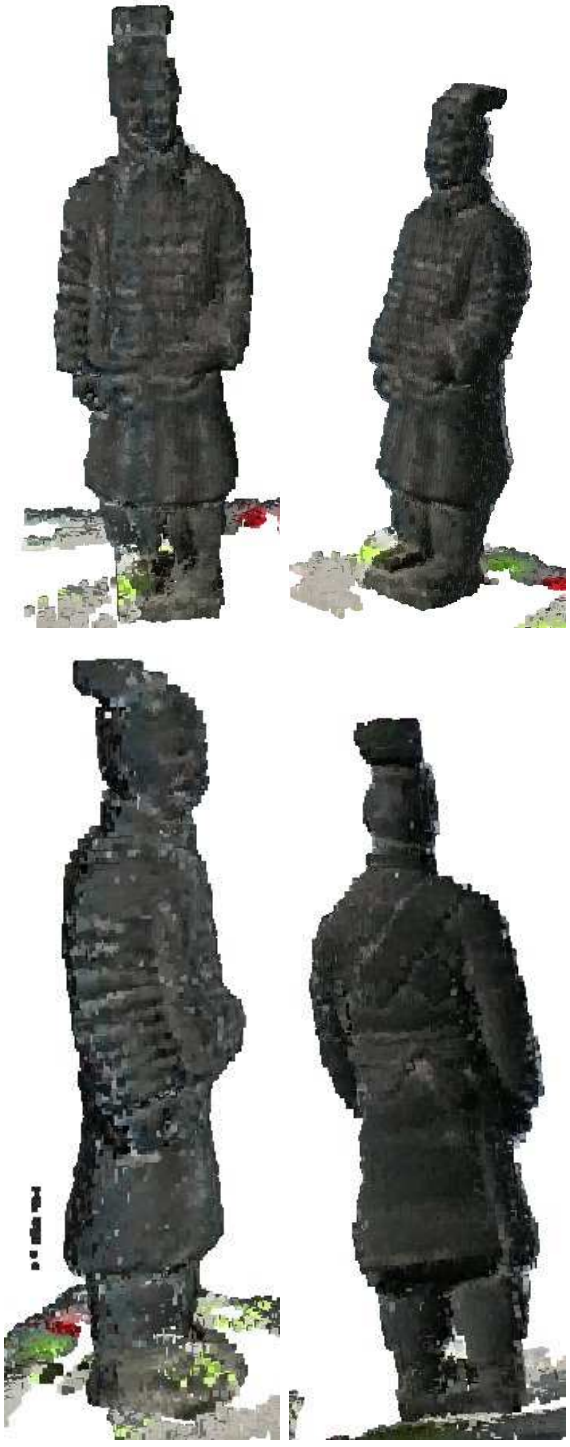

FIG. 5.11 – Modèle 3D du soldat chinois.

Chapitre 6

Conclusion

Nous avons présenté un système de reconstruction 3D interactif basé sur la sculpture d'espace composé d'équipements abordables au grand public. L'usager peut manipuler la position de la caméra vidéo autant que la position de l'objet interactivement avec un retour visuel instantané de la progression de la reconstruction. Les outils mis en place permettent à l'usager d'avoir une intuition sur les zones de l'objet plus difficiles pour l'algorithme (plus de détails de géométrie, textures, etc.).

Nous avons mis en place deux structures d'images pour répondre au grand nombre d'images entrant continuellement dans le système. Une structure statique permet d'obtenir une bonne distribution des points de vue autour de l'objet pour la comparaison de couleurs alors que l'autre structure dynamique permet de mettre l'emphase sur les images les plus récentes pour sculpter le plus de silhouettes possible.

L'utilisation d'un projecteur s'est avérée cruciale. Il facilite une calibration automatique stable et sans intervention de l'usager. Nous avons exploité en partie l'information que la lumière structurée apporte pour obtenir des points de calibration directement sur l'objet, permettant ainsi d'aller chercher des images calibrées contenant des détails de surface de l'objet. Nous avons constaté que l'utilisation du projecteur donnait une flexibilité très intéressante à notre système.

Nous avons intégré une structure hiérarchique (octree) de voxels. Cette représentation de notre modèle 3D permet une gestion de mémoire plus efficace, un traitement de la sculpture d'espace plus rapide et des modèles 3D de meilleure qualité (plus grandes résolutions). De plus, l'octree permet de traiter les images avec une résolution de grille appropriée de telle sorte que les voxels se projettent dans moins de 16 pixels. Des

images à différentes distances de l'objet peuvent ainsi être traitées correctement et simultanément. L'usager peut visualiser les différents niveaux de la hiérarchie, ce qui s'avère utile pour déterminer quelle peut être une résolution finale intéressante du modèle 3D.

Les expérimentions avec notre système et le retour visuel sont très utiles pour une bonne sélection d'images, une calibration automatique, une segmentation automatique ainsi que des initialisations de variables de système pour ensuite être utilisés lors d'une sculpture d'espace traditionnel hors-ligne.

Nous pensons que notre système procure plusieurs avantages et que les outils mis en place pour répondre à notre philosophie d'interactivité ont été bien choisis. Les résultats obtenus montrent une certaine qualité de reconstruction mais malheureusement nous n'avons pas obtenu un gain aussi significatif que ce que l'on aurait espéré par rapport à la technique originale. Par contre, nous avons un système qui facilite beaucoup la vie à l'usager pour l'acquisition, le traitement et la calibration de ces images ainsi que pour la convivialité d'utilisation.

Dans la prochaine section, nous allons discuter des améliorations possibles de notre système et des ajouts potentiellement intéressants.

6.1 Amélioration et extensions

Nous avons constaté que la boucle interactive de notre système était utile pour aider l'usager à bien initialiser les différentes parties du système. Par contre, après plusieurs utilisations, nous avons remarqué que certaines parties du processus de reconstruction devrait être plus automatisées. Par exemple, l'usager prend en général des images distribuées autour de l'objet avant de commencer à raffiner le modèle en travaillant sur des régions particulières. Ceci pourrait facilement être automatisé avec une table tournante, un bras robotisé ou les deux.

L'utilisation d'un projecteur s'est avéré très utile pour les différents modes de calibration. Par contre son utilisation introduit quelques problèmes. Les contrastes de couleur élevés en est un. Le plus gros désavantage est que l'environnement de reconstruction est limité à une certaine taille et à l'intérieur d'un environnement plus contrôlé. Du coup un des avantages de l'algorithme de sculpture d'espace, la généralité et la flexibilité par rapport aux types d'objets pouvant être reconstruits en devient quelque peu restreint.

L'utilisation de miroirs, tel qu'abordée dans le travail de Epstein *et al.* [EGPP04],

est une direction intéressante. Effectivement, grâce aux miroirs, plusieurs points de vue de l'objet peuvent être captés et traités de façon appropriée. Cependant les problèmes reliés à une illumination changeante causée par le déplacement des miroirs forment une limitation de leur usage. De plus, les couleurs captées dans les miroirs peuvent différer des couleurs captées directement par la caméra. Il faudrait par conséquent adapter la comparaison de couleurs pour ne pas éliminer des voxels valides et tenant compte de ces facteurs.

Un autre aspect que nous avons vaguement exploré mais qui mérite de l'être plus est l'étude de la réflectance des surfaces des objets. Effectivement, à partir des images acquises et de l'information qu'elles procurent, la comparaison de couleurs pourrait s'adapter au type de propriétés de réflectance de la surface. La difficulté est l'interdépendance entre la géométrie et la réflectance de la surface, c'est-à-dire qu'il est difficile d'extraire des propriétés de réflectance lorsque la géométrie est connue, alors c'est encore plus difficile lorsqu'elle ne l'est pas, et vice versa. Yang *et al.* [YPW03] ont commencé à s'attaquer à ce problème en adaptant la comparaison de couleurs pour reconstruire des surfaces spéculaires plastiques. Cependant il reste beaucoup de travail à faire dans ce domaine puisque ce ne sont pas tous les objets qui sont constitués de plastique.

Le filtrage de géométrie hiérarchique est un autre axe de recherche intéressant. Dans notre système, nous traitons correctement les niveaux de détail dans les images mais nous n'utilisons pas pleinement les informations que procurent ces différents niveaux. Par exemple, nous pourrions construire des cônes de normales, vérifier si l'information est cohérente entre deux niveaux en comparant les couleurs des enfants avec celle du parent, etc. Le modèle 3D hiérarchique pourrait alors être utilisé directement en représentation des niveaux de détail selon la distance à l'image.

Nous avons vaguement exploité un mélange de techniques différentes à l'intérieur de notre système. Nous l'avons fait avec la lumière structurée pour obtenir des points de calibration sur la surface de l'objet. Beaucoup d'autres informations auraient pu être exploitées, comme utiliser les points de la lumière structurée pour creuser dans la structure de voxels. Aussi, il est intéressant de noter que la philosophie du système peut s'appliquer à toute technique de reconstruction basée sur des images calibrées.

Annexe A

Code en C++ pour la calibration d'un modèle de caméra affine

Dans cette annexe, nous donnons le code en C++ pour calculer la matrice de calibration affine à partir de 8 correspondances 2D-3D ou plus. Le résultat de la fonction est une matrice 3×4. Cette matrice permet de trouver la projection de n'importe quel point 3D dans l'image de la caméra calibrée. Par contre, le format de cette matrice n'est pas directement compatible avec les matrices OpenGL GL_MODELVIEW et GL_PROJECTION. L'annexe qui suit donne le code pour effectuer cette conversion à partir de la matrice construite dans cette annexe. Nous utilisons la librairie mathématique GSL disponible sur LINUX. Les arguments `points` et `pixels` contiennent les correspondances entre les points 3D et 2D et sont de tailles égales. La matrice de calibration affine est retournée par la fonction.

```
#include <gsl/gsl_linalg.h>

gls_matrix*
compute_affine_matrix( ::std::vector<Point3D>& points ,
                       ::std::vector<Point3D>& pixels )
{
    if ( points.size() != pixels.size() )
    {
        return 0 ;
    }

    // Etape 1 - Construire la matrice qui contient le
    // système d'équations linéaire

    gsl_matrix* lhs = gsl_matrix_calloc( 2 * pixels.size() , 12 ) ;

    for ( int i = 0 ; i < (int)pixels.size() ; ++i )
    {
        // lignes paires
        gsl_matrix_set( lhs , 2*i , 0 , points[i].x ) ;
        gsl_matrix_set( lhs , 2*i , 1 , points[i].y ) ;
        gsl_matrix_set( lhs , 2*i , 2 , points[i].z ) ;
        gsl_matrix_set( lhs , 2*i , 3 , 1 ) ;
        gsl_matrix_set( lhs , 2*i , 4 , 0 ) ;
        gsl_matrix_set( lhs , 2*i , 5 , 0 ) ;
        gsl_matrix_set( lhs , 2*i , 6 , 0 ) ;
        gsl_matrix_set( lhs , 2*i , 7 , 0 ) ;
        gsl_matrix_set( lhs , 2*i , 8 , -pixels[i].x * points[i].x ) ;
        gsl_matrix_set( lhs , 2*i , 9 , -pixels[i].x * points[i].y ) ;
        gsl_matrix_set( lhs , 2*i , 10 , -pixels[i].x * points[i].z ) ;
        gsl_matrix_set( lhs , 2*i , 11 , -pixels[i].x ) ;
        // lignes impaires
        gsl_matrix_set( lhs , 2*i+1 , 0 , 0 ) ;
        gsl_matrix_set( lhs , 2*i+1 , 1 , 0 ) ;
        gsl_matrix_set( lhs , 2*i+1 , 2 , 0 ) ;
        gsl_matrix_set( lhs , 2*i+1 , 3 , 0 ) ;
        gsl_matrix_set( lhs , 2*i+1 , 4 , points[i].x ) ;
        gsl_matrix_set( lhs , 2*i+1 , 5 , points[i].y ) ;
        gsl_matrix_set( lhs , 2*i+1 , 6 , points[i].z ) ;
        gsl_matrix_set( lhs , 2*i+1 , 7 , 1 ) ;
        gsl_matrix_set( lhs , 2*i+1 , 8 , -pixels[i].y * points[i].x ) ;
        gsl_matrix_set( lhs , 2*i+1 , 9 , -pixels[i].y * points[i].y ) ;
        gsl_matrix_set( lhs , 2*i+1 , 10 , -pixels[i].y * points[i].z ) ;
        gsl_matrix_set( lhs , 2*i+1 , 11 , -pixels[i].y ) ;
    }
```

```
    // Etape 2 - Trouver le vecteur solution qui correspond à la plus
    // petite valeur singulière de la décomposition par valeur singulière
    // de la matrice lhs

    gsl_matrix* V = gsl_matrix_calloc( 12 , 12 ) ;
    gsl_vector* work = gsl_vector_alloc( 12 ) ;
    gsl_vector* s = gsl_vector_alloc( 12 ) ;
    gsl_vector* S = gsl_vector_alloc( 12 ) ;

    gsl_linalg_SV_decomp( lhs , V , s , work ) ;

    // Colonne contenant la plus petite valeur singulière
    int smallestCol = 11 ;

    for ( int i = 0 ; i < 12 ; ++i )
    {
        gsl_vector_set( S , i , gsl_matrix_get( V , i , smallestCol ) ) ;
    }

    // Etape 3 - Convertir le vecteur 12x1 S en
    // une matrice standard de vision 3x4 affine_matrix.

    gsl_matrix* affine_matrix = gsl_matrix_calloc( 3 , 4 ) ;

    for ( int k = 0 ; k < 12 ; ++k )
    {
        gsl_matrix_set( affine_matrix ,
                        (k / 4) ,
                        (k % 4) ,
                        gsl_vector_get( S , k ) ) ;
    }

    gsl_matrix_free( lhs ) ;
    gsl_matrix_free( V ) ;
    gsl_vector_free( work ) ;
    gsl_vector_free( s ) ;
    gsl_vector_free( S ) ;

    return affine_matrix ;
}
```

Annexe B

Code en C++ pour la conversion de matrice affine en matrices OpenGL

Cette annexe contient le code C++ pour convertir une matrice affine 3×4 en matrices compatibles avec OpenGL. Dans un premier temps, les paramètres intrinsèques de la caméra sont extraits (centre de l'image et distance focale). Ensuite, les paramètres extrinsèques sont extraits, soit la rotation et la translation de la caméra par rapport à l'origine. Le résultat de la matrice extrinsèque (GL_MODELVIEW) est retourné dans `ext_mat` pris en référence en argument de la fonction. Le résultat de la matrice intrinsèque (GL_PROJECTION) est retourné dans `int_mat` pris aussi en référence en argument de la fonction. Les arguments `width` et `height` représentent respectivement la largeur et la hauteur de l'image, en pixels.

```
#include <gsl/gsl_linalg.h>

void
extract_proj_and_modelview( Matrix3D affine_matrix ,
                            Matrix3D& ext_mat ,
                            Matrix3D& int_mat ,
                            double width ,
                            double height )
{
    double scale = affine_matrix.getRow( 2 ).length() ;
    affine_matrix = affine_matrix * (1.0 / scale) ;

    if ( affine_matrix.get( 2 , 3 ) > 0 )
    {
        affine_matrix = affine_matrix * -1 ;
    }

    Vector3D Q1 = affine_matrix.getRow( 0 ) ;
    Vector3D Q2 = affine_matrix.getRow( 1 ) ;
    Vector3D Q3 = affine_matrix.getRow( 2 ) ;

    double q14 = affine_matrix.get( 0 , 3 ) ;
    double q24 = affine_matrix.get( 1 , 3 ) ;
    double q34 = affine_matrix.get( 2 , 3 ) ;

    double tz = q34;
    double tzeps = 1;
    if ( tz > 0 )
    {
        tzeps = -1 ;
    }

    // Extraction des paramètres intrinsèques (Ox, Oy, Fx, Fy)
    // et des paramètres extrinsèques (r1, r2, r3, tx, ty, tz)
    tz = tzeps * q34 ;
    Vector3D r3 = Q3 * tzeps ;
    double Ox = dot( Q1 , Q3 ) ;
    double Oy = dot( Q2 , Q3 ) ;
    double Fx = cross( Q1 , Q3 ).length() ;
    double Fy = cross( Q2 , Q3 ).length() ;
    Vector3D r1 = (Q1 - (Q3 * Ox)) * (tzeps / Fx) ;
    Vector3D r2 = (Q2 - (Q3 * Oy)) * (tzeps / Fy) ;
    double tx = tzeps * (q14 - Ox * tz) / Fx ;
    double ty = tzeps * (q24 - Oy * tz) / Fy ;
```

```
    // Construction des matrices de rotation
    // et de translation puis de la matrice GL_MODELVIEW

    rot_mat = Matrix3D( r1.x , r1.y , r1.z , 0 ,
                        r2.x , r2.y , r2.z , 0 ,
                        r3.x , r3.y , r3.z , 0 ,
                        0 , 0 , 0 , 1 ) ;

    // Pour rendre la matrice de rotation orthonormale, il
    // faut la multiplier par la transposée de sa décomposition
    // par valeur singulière
    rot_mat = massage_orthonormal( &rot_mat ) ;
    trans_mat = Matrix3D( 1 , 0 , 0 , tx ,
                          0 , 1 , 0 , ty ,
                          0 , 0 , 1 , tz ,
                          0 , 0 , 0 , 1 ) ;
    ext_mat = trans_mat * rot_mat ;


    // Construction de la matrice GL_PROJECTION

    double far_plane = 100.0 ;
    double near_plane = 0.1 ;
    Fx = Fx / (width / 2.0) ;
    Fy = Fy / (height / 2.0) ;
    Ox = -Ox / (width / 2.0) + 1.0 ;
    Oy = -Oy / (height / 2.0) + 1.0 ;
    int_mat = Matrix3D::scale( 1 , 1 , -1 ) *
        Matrix3D( -Fx , 0 , Ox , 0 ,
                  0 , Fy , -Oy , 0 ,
                  0 , 0 , (far_plane+near_plane)/(far_plane-near_plane) ,
                  2*far_plane*near_plane/(far_plane-near_plane) ,
                  0 , 0 , -1 , 0 ) ;

}
```

Annexe C

Algorithme de comparaison de couleurs approximative

La comparaison de couleurs requiert de déterminer si deux ou plusieurs ensembles de couleurs partagent une couleur "similaire". Nous présentons dans cette annexe l'algorithme proposé par Kutulakos [Kut00] pour résoudre ce problème efficacement ; nous l'utilisons dans notre algorithme de comparaison de couleurs. Cet algorithme fonctionne pour k fenêtres (F) représentant la couverture d'un voxel dans k images. Il est à noter que les listes de couleurs des fenêtres sont triées afin d'accélérer la recherche des couleurs similaires.

- **Étape 0** : Retourner **succès** si et seulement si les étapes 1 à 4 retournent **succès** pour toutes les composantes $C = \{R, G, B\}$. Dans ce cas, assigner la couleur (μ_R, μ_G, μ_B) au voxel correspondant.

- **Étape 1** : Soit R_i, $i = 1, ..., k$ la liste des valeurs des pixels de F_i pour une composante de C.

- **Étape 2** : Trier R_1 et répéter pour chaque fenêtre F_i, $i = 2, ..., k$:

  a. trier R_i ;

  b. pour chaque valeur $R_1[j]$, trouver sa valeur la plus similaire, R_i^j, dans R_i.

- **Étape 3** : Pour chaque valeur $R_1[j]$, calculer la variance σ_j des valeurs dans l'ensemble $\{R_1[j], R_2^j, ..., R_k^j\}$.

- **Étape 4** : Si $\min_j \sigma_j < \tau$, τ étant le seuil de tolérance, retourner à l'étape 0 la moyenne, μ_C, de la valeur de la composante avec **succès d'une composante** ; sinon, retourner **échec** à l'étape 0.

Annexe D

Glossaire

- ***Ellipsoïde de couleur*** : Approxime une distribution gaussienne de couleurs dans l'espace RGB. Il est utilisé pour l'identification des points de calibration, le suivi du piédestal et la segmentation de l'arrière-plan.
- ***Environnement de reconstruction*** : Trois panneaux rectangulaires en bois, mesurés et peints en blanc mat. Cet intérieur de cube contient les objets à reconstruire et sert à la calibration du projecteur et de la caméra.
- ***Marqueur de couleur*** : petit collant de couleur rond collé sur le piédestal, facilitant l'identification dans la caméra.
- ***Modèle 3D*** : Résultat de la numérisation de l'objet.
- ***Objet*** : Objet réel à reconstruire.
- ***Piédestal*** : Petite boîte rectangulaire sur lequel l'objet est placé lors de la reconstruction. Il est suivi dans la caméra et permet de déterminer où se trouve l'objet dans l'environnement de reconstruction.
- ***Point de calibration*** : Point de couleur projeté par le projecteur sur l'environnement de reconstruction afin de faciliter la calibration automatique de la caméra. Seuls les rouge, vert et bleu saturés et le gris sont utilisés afin de prévenir les limitations de la caméra.
- ***Sculpture d'espace*** : *Space carving.*
- ***Usager*** : Personne qui interagit avec notre système de reconstruction.

Bibliographie

[Aya91] N. Ayache. *Artificial Vision for Mobile Robots – Stereo-vision and Multi-sensor Perception*. MIT Press, 1991.

[BBM+01] C. Buehler, M. Bosse, L. McMillan, S.J. Gortler et M.F. Cohen. « Unstructured Lumigraph Rendering ». Dans *Proc. SIGGRAPH 2001*, pages 425–432, août 2001.

[Bou] Boujou. « www.2d3.com ».

[BR00] F. Bernardini et H. Rushmeier. « The 3D Model Acquisition Pipeline ». Dans *Eurographics 2000 : STAR*, septembre 2000.

[Can] Canoma. « www.canoma.com ».

[CM99] Q. Chen et G. Medioni. « A Volumetric Stereo Matching Method : Application to Image-based Modeling ». Dans *Proc. Computer Vision and Pattern Recognition*, pages 29–34, juin 1999.

[DA89] U.R. Dhong et J.K. Aggarwal. « Structure from Stereo — A Review ». *IEEE Trans. on Systems, Man, and Cybernetics*, volume 19, numéro 6, pages 1489–1510, 1989.

[Ded01] S. Dedieu. *Adaptation of a 3D Model Reconstruction System from Photos to User Knowledge*. Thèse de doctorat, Université Bordeaux I, décembre 2001.

[DTM96] P.E. Debevec, C.J. Taylor et J. Malik. « Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-Based Approach ». Dans *Proc. SIGGRAPH 96*, pages 11–20, août 1996.

[EGPP04] E. Epstein, M. Granger-Piché et P. Poulin. « Exploiting Mirrors in Interactive Reconstruction with Structured Light ». Dans *Vision, Modeling, and Visualization 2004*, pages 125–132, novembre 2004.

[Eps05] E. Epstein. « Utilisation de miroirs dans un système de reconstruction interactif ». Mémoire de maîtrise, Université de Montréal, janvier 2005.

[Fau93] O. Faugeras. *Three-Dimensional Computer Vision — A Geometric Viewpoint*. MIT Press, 1993.

[GGSC96] S.J. Gortler, R. Grzeszczuk, R. Szeliski et M.F. Cohen. « The Lumigraph ». Dans *Proc. SIGGRAPH 96*, pages 43–54, août 1996.

[Gla89] A. Glassner. *An Introduction to Ray Tracing*. Academic Press Limited, 1989.

[GP04] M. Granger-Piché. « www.iro.umontreal.ca/labs/infographie/theses/grangem/ », 2004.

[GPEP04] M. Granger-Piché, E. Epstein et P. Poulin. « Interactive Hierarchical Space Carving with Projector-based Calibrations ». Dans *Vision, Modeling, and Visualization 2004*, pages 159–166, novembre 2004.

[KS00] K.N. Kutulakos et S.M. Seitz. « A Theory of Shape by Space Carving ». *International Journal of Computer Vision*, volume 38, numéro 3, pages 199–218, 2000.

[KSK98] R. Klette, K. Schluns et A. Koschan. *Computer Vision : Three-Dimensional Data from Images*. Springer, 1998.

[Kut00] K.N. Kutulakos. « Approximate N-View Stereo ». Dans *European Conference on Computer Vision*, pages 67–83, 2000.

[LH96] M. Levoy et P.M. Hanrahan. « Light Field Rendering ». Dans *Proc. SIGGRAPH 96*, pages 31–42, août 1996.

[MBR+00] W. Matusik, C. Buehler, R. Raskar, S. Gortler et L. McMillan. « Image-based Visual Hulls ». Dans *Proc. SIGGRAPH 2000*, pages 369–374, juillet 2000.

[MPN+02] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler et L. McMillan. « Image-based 3D Photography using Opacity Hulls ». *ACM Trans. on Graphics*, volume 21, numéro 3, pages 427–437, juillet 2002.

[PDH+97] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro et W. Stuetzle. « Robust meshes from multiple range maps », 1997.

[PFTV92] W.H. Press, B.P. Flannery, S.A. Teukolsky et W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1992.

[Pho] Photomodeler. « www.photomodeler.com ».

[POF98] P. Poulin, M. Ouimet et M.-C. Frasson. « Interactively Modeling with Photogrammetry ». Dans *Eurographics Workshop on Rendering 1998*, pages 93–104, juin 1998.

[PSD+03] P. Poulin, M. Stamminger, F. Duranleau, M.-C. Frasson et G. Drettakis. « Interactive Point-Based Modeling of Complex Objects from Images ». Dans *Graphics Interface 2003*, pages 11–20, juin 2003.

[Ras03a] R. Raskar. « Camera Calibration ». *SIGGRAPH Course Notes 21*, juillet 2003.

[Ras03b] R. Raskar. « www.merl.com/people/raskar/raskar.html », 2003.

[RCM+01] C. Rocchini, P. Cignoni, C. Montani, P. Pingi et R. Scopigno. « A Low Cost 3D Scanner based on Structured Light ». *Computer Graphics Forum*, volume 20, numéro 3, pages 299–308, 2001.

[Rea] RealViz. « www.realviz.com ».

[RHHL02] S. Rusinkiewicz, O. Hall-Holt et M. Levoy. « Real-Time 3D Model Acquisition ». *ACM Trans. on Graphics*, volume 21, numéro 3, pages 438–446, juillet 2002.

[RL00] S. Rusinkiewicz et M. Levoy. « QSplat : A Multiresolution Point Rendering System for Large Meshes ». Dans *Proc. SIGGRAPH 2000*, pages 343–352, juillet 2000.

[SD99] S.M. Seitz et C.R. Dyer. « Photorealistic Scene Reconstruction by Voxel Coloring ». *International Journal of Computer Vision*, volume 35, numéro 2, pages 151–173, 1999.

[SG99] R. Szeliski et P. Golland. « Stereo Matching with Transparency and Matting ». *International Journal of Computer Vision*, volume 32, numéro 1, pages 45–61, août 1999.

[Tsa87] R. Tsai. « A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-shelf TV Cameras and Lenses ».

IEEE Robotics and Automation, volume 3, numéro 4, pages 323–344, août 1987.

[TV98] E. Trucco et A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[YPW03] R. Yang, M. Pollefeys et G. Welch. « Dealing with Textureless Regions and Specular Highlights – A Progressive Space Carving Scheme Using a Novel Photo-consistency Measure ». Dans *Proc. Int. Conf. on Computer Vision*, pages 576–584, juillet 2003.