Modélisation tridimensionnelle à partir de silhouettes

Jean-Sébastien Franco

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE
pour obtenir le grade de
DOCTEUR DE L'INPG

Spécialité : "Imagerie, Vision, Robotique"

préparée au laboratoire Gravir - Imag - Inria Rhône-Alpes
dans le cadre de **l'Ecole Doctorale *"Mathématiques, Sciences et Technologies de l'Information, Informatique"***

présentée et soutenue publiquement par

Jean-Sébastien FRANCO

le 13 Décembre 2005

Titre :

MODÉLISATION TRIDIMENSIONNELLE À PARTIR DE SILHOUETTES

Directeurs de Thèse :
Edmond Boyer, Radu Horaud

JURY

M.	**Jean-Claude LÉON**	Président
M.	**Marc POLLEFEYS**	Rapporteur
M.	**Jean PONCE**	Rapporteur
M.	**Radu HORAUD**	Directeur de thèse
M.	**Edmond BOYER**	Co-directeur de thèse
M.	**Jean-Marc HASENFRATZ**	Examinateur

A la mémoire de Frederick A. Franco et Vincent Franco

Remerciements

Je souhaite remercier Jean-Claude Léon, pour avoir accepté de présider mon jury de thèse. Je souhaite remercier particulièrement Radu Horaud et toute l'équipe MOVI pour m'avoir accueilli dans d'excellentes conditions.

Un grand merci à Jean Ponce et Marc Pollefeys, pour avoir porté un intérêt tout particulier à mon travail en acceptant d'être rapporteur de cette thèse. Merci également à Jean-Marc Hasenfratz pour sa participation au Jury et pour son rôle déterminant dans mes débuts à l'INRIA et les projets CYBER.

Je souhaite tout particulièrement remercier Edmond Boyer qui a été un excellent directeur de thèse, qui a su me guider et me témoigner sa confiance à des moments clés de ce parcours semé d'embûches et de doutes.

Merci bien sûr à Clément Ménier, Jérémie Allard et Bruno Raffin pour leur collaboration fructueuse et passionnée au sein de la plateforme GrImage et pour les nombreuses heures de labeur communes à toute heure de la journée ou de la nuit, aboutissant à des résultats moteurs de cette thèse.

Merci également à Pierre Bessière et Peter Sturm pour les discussions édifiantes et constructives dont cette thèse a clairement bénéficié.

J'adresse aussi une pensée très amicale à tous ceux qui ont partagé mon quotidien, en particulier mes collègues de bureau successifs, qui ont tous été très bon public pour mes facéties diverses, dans une ambiance très sympathique : Adrien Bartoli, Thomas Bonfort, Aude Jacquot, Andrei Zaharescu, sans oublier bien sûr tous les thésards et permanents du laboratoire, dont la liste serait trop longue mais que je n'oublie pas, pour les échanges et les bons moments.

Enfin j'adresse ma plus profonde et chaleureuse gratitude à Stéphanie, à mes parents, et à ma famille, qui ont tous joués un rôle déterminant dans le développement de ma curiosté et de ma passion pour la science, de par leur confiance et leur soutien inconditionnel.

Résumé

Nous nous intéressons dans cette thèse à la modélisation de formes tridimensionnelles à partir de vidéos numériques. Plus particulièrement nous considérons les approches utilisant des silhouettes, acquises à l'aide de caméras calibrées. Notre objectif est de fournir une modélisation rapide pour permettre au modèle construit d'interagir en temps réel avec des environnements virtuels. Parmi les différentes primitives qui peuvent être considérées pour acquérir des modèles tridimensionnels, les silhouettes offrent l'avantage de simplifier la gestion de la visibilité et de permettre des traitements rapides. Les principales limitations des méthodes qui en découlent concernent le rapport défavorable entre la précision du modèle construit et le temps de calcul nécessaire, ainsi que l'extraction des silhouettes dans les images qui reste une étape sensible du processus de modélisation. En réponse à ces limitations, nous proposons deux approches rapides de modélisation à partir de silhouettes. La première suppose que les données, les silhouettes et le calibrage, sont précises, et produit un modèle exact vis-à-vis de ces données. La deuxième permet d'introduire des incertitudes dans les primitives considérées et produit alors un modèle probabiliste du volume observé. Ces deux approches sont illustrées par des résultats sur données synthétiques et réelles, acquises sur la plate-forme GrImage de l'Inria Rhône-Alpes. En particulier une implémentation distribuée et temps réel de la première méthode est proposée. Celle-ci est validée dans le contexte d'une application de réalité virtuelle.

Mots-clés : Modélisation à partir d'images, modélisation à partir de silhouettes, reconstruction 3D, enveloppe visuelle, fusion multi-vue.

Abstract

This thesis focuses on the problem of tri-dimensional shape modeling from digital video streams. In particular we consider approaches which use silhouettes, acquired using calibrated cameras. Our goal is to provide fast modeling methods : the models produced can then be used for real-time interaction in virtual environments. Among the different primitives which can be used for 3D model acquisition, silhouettes have several advantages, as they simplify the handling of visibility, and can usually be processed rapidly. Approaches which use silhouettes are generally subject to limitations, such as a poor trade-off between model precision and processing time, and automatic silhouette extraction which can be corrupted by various types of noise. In response to these limitations, we propose two fast approaches for shape modeling from silhouettes. The first approach assumes that the input data, the silhouettes and calibration, are reliable, and produces an exact model with respect to this data. The second approach deals with noisy inputs and produces a probabilistic model of the observed shape's volume. We provide results for both approaches, using both synthetic and real input sets, acquired on the GrImage platform at the Inria Rhône-Alpes. In particular we propose a distributed real-time implementation of the first approach. The system is demonstrated and validated in the context of a virtual reality application.

Keywords : Modeling from multiple views, modeling from silhouettes, shape-from-silhouettes, 3D reconstruction, visuall hull, multi-view fusion.

Table des matières

1.

Introduction

La communauté scientifique s'intéresse de près à toute initiative permettant de rendre les machines plus intelligentes, et d'en faire des entités perceptives capables d'acquérir de l'information à partir d'un environnement réel, de la traiter et de l'interpréter. Plusieurs disciplines de l'informatique étroitement liées sont nées autour de ce paradigme, comme l'intelligence artificielle, la robotique, la vision par ordinateur, avec l'espoir de rendre les machines plus autonomes et plus utiles, en leur faisant exécuter des tâches de plus en plus complexes.

Dans cette optique, tout processus de vision par ordinateur se base spécifiquement sur les images acquises d'une scène pour en déduire un état. Les données nécessaires à un tel processus sont produites à l'aide d'une ou plusieurs caméras numériques. Les tâches rentrant dans ce champ de définition peuvent être de nature très diverse : reconnaissance d'objets ou de formes, classification, suivi d'objet en mouvement, reconstitution de la géométrie de la scène, qu'il s'agisse de la forme des objets observés, la position des caméras elles-mêmes, ou la résolution simultanée de plusieurs de ces problèmes.

Nous nous penchons dans le cadre de cette thèse sur la modélisation de la forme des objets dans une scène dynamique à partir d'images, et en particulier à partir des silhouettes d'objets dans les images. Il s'agit donc d'estimer un modèle géométrique du contenu de celle-ci, c'est à dire une information 3D, à partir d'images 2D de la scène. Ce type d'information est particulièrement utile pour toute application nécessitant d'acquérir, de traiter, ou d'afficher des modèles 3D synthétiques à partir d'objets réels (acquisition 3D, synthèse d'images, capture de mouvement, etc.). Nous nous intéressons donc ici davantage à la structure géométrique de la scène plutôt qu'à son contenu sémantique, bien que les modèles 3D que nous construisons peuvent eux-mêmes servir à des applications de reconnaissance. Motivés par les évolutions récentes en matière de technologie

d'acquisition vidéo et en informatique, où la fabrication de plates-formes comportant plusieurs ordinateurs et caméras devient courante et peu onéreuse, nous choisissons de traiter le problème dans un contexte multi-caméras.

Les progrès continus sur les ordinateurs eux-mêmes, et leur gain constant en puissance, ouvrent les possibilités d'exécution de telles tâches complexes en un temps de plus en plus court, et parfois de l'ordre très inférieur à la seconde, ce qui était impensable il y a seulement quelques années. C'est donc tout naturellement que l'on peut aujourd'hui insérer de telles tâches de modélisation dans des environnements interactifs, des boucles de contrôles, ou tout autre contexte temps réel. Nous illustrerons ces possibilités dans le cadre de cette thèse, en appliquant les résultats obtenus à une famille d'approches interactives appelée *réalité virtuelle*, où l'on met en rapport et en interaction monde réel et mondes synthétisés par l'ordinateur. Les recherches dans ces thématiques sont motivées par la grande quantité d'applications directes qu'elles comportent : interaction vidéoludique, audiovisuel, réalité augmentée, opérations et contrôle à distance, visioconférence enrichie, extraction automatique de modèles, aide à la conception ergonomique, simulations, et autres débouchés très divers. Bien sûr, la réalité virtuelle ne constitue qu'un champ d'application possible pour les méthodes de modélisation à partir d'images, et les résultats de cette thèse peuvent être appliqués à un éventail plus large de problèmes.

Il existe de nombreuses familles d'approches pour modéliser des formes à partir d'images (*shape from X*). Certaines méthodes n'utilisent qu'une seule image, à condition de disposer d'information et de connaissances a priori supplémentaires. L'information apportée par une seule image 2D ne peut sinon aboutir à la construction d'un modèle 3D, si rien n'est fait pour lever l'ambiguïté fondamentale de projection due à la perte de l'information de profondeur. Parmi celles utilisées dans les méthodes existantes, il y a les propriétés de réflectance d'une surface (*shape from shading*) [84], la texture des surfaces (*shape from texture*) [58]. Dès que plusieurs images d'une même scène sont disponibles, divers mécanismes de détection de cohérence et de triangulation permettent d'apporter une réponse géométrique à la question du contenu de la scène. Il est par exemple possible, à partir des disparités entre niveaux de gris de deux images prises de points de vue très proches, de retrouver l'information de profondeur pour chaque pixel de ces images (*shape from stereo*) [10].

Beaucoup de méthodes cherchent à éliminer les zones de l'espace où il n'y pas de matière, en utilisant l'information des images pour détecter les régions incohérentes de l'espace. A cette fin certaines méthodes utilisent l'information photométrique pour trouver une représentation volumétrique de la scène (*space carving* [62], *voxel coloring* [90]). D'autres utilisent dans ce but les silhouettes d'ob-

jets dans les images [11, 98, 22] (*shape from silhouettes*), ou plus spécifiquement les contours des objets (*shape from contours*) [17, 100].

Nous avons choisi d'étudier plus en détail tout au long de cette thèse, quelle forme il est possible d'obtenir à partir des silhouettes. En effet les silhouettes constituent un moyen fiable de définir une partition de l'image en régions, qui induisent une forme dans l'espace, l'*enveloppe visuelle* [64]. Cette forme, même si elle est une approximation de la véritable forme de l'objet, présente de très bonnes propriétés : celle de contenir la forme réelle de l'objet, de pouvoir être calculée rapidement, et de converger vers la forme de l'objet privée de ses concavités lorsque l'on augmente le nombre de points de vues. Un certain nombre de travaux existent déjà pour calculer une forme à partir de silhouettes, optant pour un approche volumétrique ou surfacique, selon que l'on s'intéresse au volume de l'enveloppe visuelle ou à sa surface délimitante. Nous reviendrons en détail sur ces méthodes au chapitre 3.

Formulation du problème

Nous proposons dans cette thèse des solutions au problème de modélisation de formes à partir de silhouettes. Nous supposons que N caméras observent une scène avec des objets en mouvement, et qu'elles sont calibrées, c'est à dire que leur position et réglages sont connus. L'acquisition de N images est effectuée à intervalle fixe en vue de leur traitement, pour reconstruire la forme géométrique des objets de la scène à chaque instant. Aucune supposition n'est faite *a priori* sur la nature des formes à reconstruire : nous nous plaçons de ce fait dans un contexte très général.

L'approche classique, dont le déroulement est schématisé en figure 1.1 consiste à segmenter chaque images en deux régions, le fond d'une part, et les silhouettes des objets à reconstruire d'autre part. Pour les besoins du traitement en ligne, il est donc dans ce cadre fait appel à un processus d'extraction automatique de silhouette. Il s'agit souvent en pratique d'effectuer des tests par pixel dans les images pour déterminer s'ils appartiennent au fond ou à l'objet, en s'appuyant sur un modèle d'apparence du fond. Ce modèle peut être de diverse nature comme nous le détaillerons au chapitre 2, et nécessite le plus souvent que chaque caméra ait préalablement observé la scène dépourvue de tout objet à reconstruire. Une fois les silhouettes extraites, la forme des objets est alors reconstruite par déduction géométrique à partir des silhouettes. Nous suivrons ce schéma pour les méthodes de modélisation classique que nous proposons, et montrerons une méthode alternative pour intégrer les informations provenant des silhouettes dans les images.

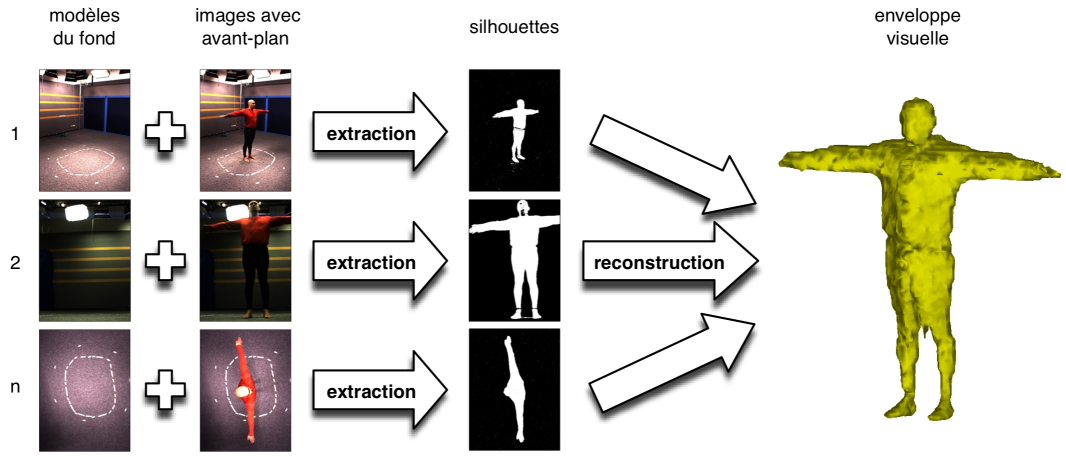


FIG. 1.1 – Stratification classique du problème de modélisation de forme à partir de silhouettes.

Difficultés et contributions

Nous étudions tout au long de cette thèse l'ensemble des traitements associés à un processus de modélisation de forme à partir de silhouettes, dans un contexte dynamique. Nous identifions un certain nombre de problèmes, et proposons tout au long de cette étude plusieurs méthodologies et solutions. Quatre nouvelles méthodes sont proposées, implémentées et validées expérimentalement, et offrent de nouvelles perspectives sur le problème de modélisation à partir de silhouettes. Deux de ces approches concernent la modélisation à partir de silhouettes au sens purement géométrique, une autre en propose une formulation statistique sous forme de fusion de capteurs, une dernière enfin est réalisée selon une stratégie distribuée, dans un contexte de calcul parallèle.

Les difficultés sont nombreuses et touchent plusieurs étapes de ce schéma, mais pour chacune d'elles nous apportons une réflexion et des contributions significatives :

– La modélisation de solides à partir de silhouettes peut donner lieu à de nombreuses formulations plus ou moins adaptées. Nous prendrons le parti de modéliser les objets en nous intéressant à leur surface, par opposition à une représentation volumétrique. Cette approche comporte en effet un certain nombre d'avantages, particulièrement la précision qu'elle permet d'atteindre dans la modélisation. Il s'agit en outre d'une approche dont les résultats sont plus adaptés au rendu graphique. Une telle représentation sous forme de surface est difficile à construire. Pour répondre à des critères

exigeants de modélisation, par opposition à une simple reconstruction approximative, il faut trouver une représentation pratique de la surface, s'assurer de produire une surface sans auto-intersection ou anomalie topologique. Nous verrons au chapitre 3 que peu de méthodes de modélisation répondent à ces critères en vision par ordinateur, et proposerons deux nouvelles méthodes de reconstruction de surface à partir de silhouettes à base de maillages polyédriques, permettant d'atteindre de tels buts. La première est une approche préliminaire, qui se sert de la triangulation de Delaunay de points sur la surface de l'enveloppe visuelle pour en proposer une reconstruction polyédrique [16, 38]. La deuxième est un algorithme de reconstruction exacte du polyèdre de l'enveloppe visuelle à partir de silhouettes polygonales [39], que nous confronterons aux approches comparables en vision et géométrie algorithmique.

– L'extraction automatique des silhouettes est un problème difficile, dont les solutions souffrent du bruit de mesure dans les images, des changements de géométrie et de luminosité de la scène pouvant survenir après observation du fond, et d'ambiguïtés dans l'espace de couleur. Toute erreur commise à cette étape corrompt les résultats de modélisations produits en aval : tout modèle reconstruit à partir de ces entrées bruitées hérite géométriquement de ces imperfections. Malheureusement peu de méthodes d'extractions de silhouettes existantes sont robustes à tous types de bruits, il faut donc admettre que ce problème est persistant et que tout algorithme de reconstruction automatique à partir d'images y sera confronté. Pour ne pas aggraver ce type de problèmes, nous montrerons au chapitre 2 comment obtenir de manière robuste des silhouettes polygonales à partir de silhouettes pixellisées pour les besoins de la modélisation, sans introduire d'erreur supplémentaire dans la chaîne de traitement. Pour améliorer la robustesse de la modélisation à ce type de problèmes, nous proposerons également au chapitre 4 une nouvelle formulation statistique de la modélisation de volumes à partir de silhouettes [40, 41], en réalisant une fusion d'informations des silhouettes, comme schématisé en figure 1.2.

– La prise en compte de la cohérence temporelle au sein du flux d'information est intrinsèquement difficile, car elle nécessite de comprendre et modéliser la dynamique de la scène, ce qui est considérablement plus compliqué que la résolution du problème statique lorsque celle-ci est constituée d'objets déformables. Il s'agit là d'une information pourtant fondamentalement utile. Le déplacement des objets entre deux acquisitions successives est contraint par la nature du mouvement dans la scène, et sa modélisation peut permettre d'affiner nos reconstructions. Certaines méthodes montrent

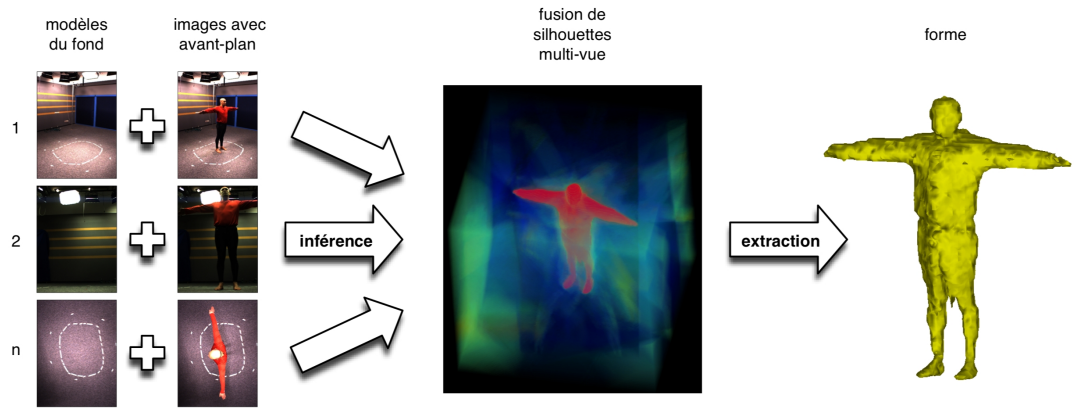


FIG. 1.2 – Nouvelle stratification proposée.

par exemple qu'un alignement des parties rigides d'objets en mouvement est possible et permet d'affiner l'enveloppe visuelle d'un objet au cours du temps [21]. D'autres posent le problème comme celui de l'estimation d'une seule forme 4D lisse rendant compte de l'information image de toute la séquence [46]. Nous verrons au chapitre 4 que le cadre des grilles d'occupation [33] ouvre une nouvelle perspective pour exploiter la cohérence temporelle au sein de la séquence.

– La vision multi-caméras et la modélisation à partir d'images nécessitent de manipuler d'importants flux de données. Le simple fait d'effectuer une acquisition d'images couleurs à 30Hz avec une dizaine de caméras implique de traiter des débits d'image de l'ordre de 10Go/minute, qu'aucun ordinateur standard isolé ne peut traiter ni même physiquement recevoir à ce jour, avant même de parler de reconstruction. Dans une optique de temps réel, cela induit l'impossibilité d'un traitement centralisé des données et impose de se pencher sur le problème du traitement distribué de l'acquisition et de la modélisation. Le traitement hors-ligne permet de s'affranchir de certaines de ces contraintes, mais ne permet d'éviter l'usage de plusieurs machines pour l'acquisition d'un grand nombre de vues synchronisées. Si un certain nombre de travaux se penchent sur la gestion distribuée de l'acquisition vidéo [14, 37, 91], peu traitent de la distribution du processus de vision lui-même. Nous proposerons une réflexion sur le calcul parallèle pour la reconstruction, et sa concrétisation sous la forme d'un algorithme de modélisation exacte distribuée à partir de silhouettes, au chapitre 5. Cette approche est validée expérimentalement dans le contexte d'une application interactive de réalité mixte [76, 42, 77, 79].

Organisation de la thèse

Ce document comporte quatre chapitres.

- Le chapitre 2 présente l'extraction de silhouettes à partir d'images. Les divers familles d'approches existantes y sont présentées, et plusieurs modèles pour l'apparence du fond abordés. Nous discutons en outre des différentes représentations possibles pour une silhouette. Parmi celles-ci, nous nous attardons sur la représentation polygonale des contours de silhouettes qui est particulièrement utile pour les méthodes de modélisation à partir de silhouettes présentées au chapitre 3. Nous montrons comment celle-ci peut être obtenue à partir de cartes d'appartenance à une silhouette, sans introduction d'erreur supplémentaire dans le traitement global.
- Le chapitre 3 présente nos contributions en matière de reconstruction géométrique à partir de silhouettes polygonales. Un état de l'art des méthodes de modélisation de forme à partir de silhouettes y est conduit, et divers rapprochements avec les travaux de géométrie algorithmique et d'image de synthèse y sont soulignés. Nous présentons deux algorithmes de modélisation de formes à partir de silhouettes polygonales, l'algorithme hybride et l'algorithme polyédrique exact, avant de comparer cette dernière approche aux méthodes de référence des divers domaines concernés.
- Le troisième chapitre présente une vue alternative du problème de modélisation à partir de silhouettes, en proposant sa formulation sous la forme d'une estimation probabiliste d'une grille d'occupation de la matière dans l'espace et d'une fusion de capteurs, s'inspirant des méthodes utilisées en robotique. En particulier dans la méthode proposée, aucune décision explicite n'est prise sur la localisation des silhouettes dans les images. Une méthodologie statistique est rigoureusement appliquée et permet de s'affranchir de plusieurs limitations propres aux approches purement géométriques, tout en permettant une meilleure robustesse au bruit et la levée sur les contraintes de visibilité commune des objets dont souffrent certaines méthodes de reconstruction classiques. Une possibilité d'extension de la méthode proposée est explorée, et ouvre des perspectives pour exploiter la cohérence temporelle de la scène.
- Le quatrième chapitre s'intéressera davantage à la réalisation temps réel de telles modélisations de forme en s'intéressant au traitement parallèle dans le cadre de la vision et de la reconstruction. Une réflexion sur les méthodes distribuées et leur application aux algorithmes de vision y est conduite. Un algorithme de reconstruction polyédrique exact distribué est proposé, discuté et expérimentalement validé dans le contexte d'une application interactive, illustrant le potentiel des méthodes abordées pour la réalité virtuelle et autres applications temps réel.

2.

Extraction de silhouettes

Toute méthode de reconstruction à partir d'images est basée sur une ou plusieurs manières de caractériser et réduire l'information présente dans celles-ci. Points appariés, droites, couleurs, segmentations en régions, constituent les briques de bases possibles sur lesquels les algorithmes de reconstruction existants sont construits. Dans le cadre de cette thèse nous nous intéressons de près à la caractérisation d'objets par leur silhouettes dans les images. Si la notion de silhouette est bien définie, le fait d'automatiser son extraction dans les images pose un problème à part entière. En effet nos données d'entrées, les images, ne sont rien de plus qu'un ensemble de grilles discrètes de couleurs et de contrastes, soumises à un bruit de mesure, dont il faut extraire une information fiable et plus abstraite : les silhouettes d'objets d'intérêt. Ce problème difficile d'extraction et son influence fondamentale sur le problème de reconstruction que nous nous posons en aval ne peuvent être ignorés. Nous examinons dans ce chapitre les différentes définitions, représentations et méthodes d'extraction possibles d'une silhouette proposées dans les travaux existants. Nous proposerons par ailleurs une première contribution : celle de caractériser les objets de la scène par leur silhouette donnée sous forme polygonale, dont l'extraction peut se faire sans perte d'information à partir d'une silhouette binaire avec un algorithme que nous explicitons. Les représentations et méthodes discutées sont les bases et les données d'entrées nécessaires aux algorithmes de reconstructions que nous présenterons dans les chapitres suivants, où figurent les principales contributions de cette thèse.

2.1 Introduction

Le problème d'extraction de silhouettes à partir d'images est difficile, et mobilise un effort de recherche important. Il est nécessaire de l'aborder dans le cadre de cette thèse, puisque nos méthodes de modélisation reposent sur l'obtention automatique de silhouettes. Ce problème peut donner lieu aux formulations les plus diverses : reconnaissance de couleurs, détection de mouvement, segmentation sur des critères divers, identification et suivi de régions ou de contours dans une séquence d'images. La méthode utilisée pour extraire automatiquement une silhouette a évidemment une influence sur la modélisation de forme que l'on réalise en aval de cette extraction. Il est donc important d'aborder les différentes méthodes existantes pour obtenir des silhouettes, et leurs limitations. Il s'agit de ne pas perdre de vue que les silhouettes sont une représentation intermédiaire utile, mais que le point d'entrée de nos algorithmes reste les images. Ce faisant nous tentons de garder à l'esprit le processus de modélisation à partir d'images dans son ensemble, pour conserver un regard critique sur le problème de reconstruction à partir de silhouettes et en élargir les perspectives. Nous confronterons en effet dans les deux chapitres suivants deux visions très différentes mais complémentaires de ce problème.

Ces éléments font l'objet du présent chapitre de thèse. Il s'agira donc pour nous de définir ce qu'est une silhouette et de poser le problème de son extraction. Nous considérons ensuite un échantillon des différentes méthodes d'extraction et représentations possibles pour celle-ci. Dans l'un des points de vue adoptés sur le problème de reconstruction à partir de silhouettes, nous nous intéressons au raisonnement purement géométrique et montrons l'information qu'il est possible d'obtenir sur la forme des objets à partir d'une représentation polygonale de leur silhouette. Nous montrons donc également dans ce chapitre, comment obtenir de telles silhouettes polygonales à partir d'une représentation plus couramment obtenue après extraction, la silhouette binaire.

2.2 Contexte et définitions

Considérons une scène, comportant plusieurs objets, observée par un ensemble de caméras. Nous ne faisons pour l'instant que très peu de suppositions sur le modèle de caméra utilisé. Considérons simplement chaque caméra en tant qu'appareil d'observation, doté d'un système optique captant un ensemble de rayons lumineux provenant de la scène, et comportant un plan image où ces rayons, et l'information de luminosité et de couleur qu'ils véhiculent, sont projetés.

Chaque caméra engendre une image bidimensionnelle de la scène, où les objets figurent. Nous nous intéressons à un sous-ensemble d'*objets d'intérêt* de la scène, les objets dont nous souhaitons reconstruire la forme, par opposition aux objets

du reste de la scène, informellement qualifiés de *fond*. Un ensemble de rayons provenant des objets d'intérêt traverse le système optique de la caméra et intersecte le plan image. La fermeture de cet ensemble de rayons engendre des régions du plan image, comportant potentiellement plusieurs composantes, où se forment l'image de ces objets. Ces régions forment la *silhouette* des objets d'intérêt. Par convention, nous utiliserons toujours le terme de silhouette au singulier pour qualifier les régions de l'espace image d'une seule et même vue, où se projettent les objets d'intérêt.

Il est possible de ne considérer les images observées qu'à un instant donné, mais il est également possible de travailler sur une plage de temps. Dans ce dernier cas, le processus d'observation décrit est continu dans le temps ; toutefois un échantillonnage de l'information est opéré aussi bien dans le temps (acquisition à intervalle fixe) que dans l'espace (pixels du capteur CCD des caméras) pour son traitement numérique.

Se pose alors le problème d'extraction des silhouettes d'objets d'intérêt dans chaque image à partir de la seule information dont on dispose : la couleur en chaque pixel. Selon l'approche choisie, celle-ci peut être exploitée directement ou faire l'objet de pré-traitements locaux supplémentaires pour en extraire certaines caractéristiques : détection de bords dans les images, estimation d'un mouvement dans une séquence d'images. Pour la plupart des méthodes d'extraction, il est nécessaire de rajouter certaines hypothèses : celle d'un fond statique (ou seuls les objets d'intérêt sont en mouvement), et celle de caméras statiques, c'est à dire avec des caméras immobiles aux paramètres fixes.

2.3 Méthodes d'extraction

Dans le but d'éclairer le lecteur sur la nature du problème d'extraction et ses solutions, nous donnons ici un échantillon des techniques existantes pour effectuer cette tâche, représentatives mais non exhaustives.

2.3.1 Approches pixel

La forme discrète des images rend naturelle la réalisation d'un traitement par pixel pour extraire les silhouettes. En conséquence la majorité des méthodes d'extraction de silhouettes réalisent des caractérisations par pixel : on cherche dans ce cas à qualifier l'appartenance ou non de chaque pixel à la silhouette. Voici quelques méthodes classiques rentrant dans cette catégorie :

– Le **chroma keying** est une technique d'extraction de silhouette très utilisée dans l'audiovisuel. Elle nécessite un environnement très conditionné, d'une couleur matte uniforme (généralement bleu ou vert). Cette méthode repose

sur l'estimation pour chaque pixel d'un paramètre de transparence (alpha) en fonction de la distance de la couleur de chaque pixel à la couleur de référence. La méthode engendre donc une carte de transparence (ou *alpha-map*), en tant que représentation de la silhouette. Le chroma keying est une technique très fiable mais généralement appliquée dans des conditions très particulières, avec un environnement à la lumière contrôlée, dans un studio d'acquisition.

– La **différence de couleur** suppose une scène fixe et la connaissance d'une image de la scène dépourvue d'objet d'intérêt (ou image du fond). Les couleurs des pixels de l'image de travail sont alors comparées directement avec les couleurs observées du fond en ces mêmes pixels, en estimant une distance entre les deux couleurs. Dans sa variante la plus simple, un seuil est fixé sur cette distance pour séparer les couleurs du fond et de l'avant plan. Ce seuillage engendre donc une silhouette sous la forme d'une carte d'étiquetage, que l'on appellera par la suite *silhouette binaire*, où chaque pixel peut être marqué de deux et seulement deux façons, selon s'il appartient ou non à la silhouette. On peut aussi estimer un paramètre de transparence alpha en chaque pixel, en se basant sur ces informations. Ces techniques nécessitent un fond complètement statique et est très peu robuste aux différentes variations pouvant intervenir dans les images (bruit capteur, mouvement d'objets du fond, changements de luminosité, etc.). Elle n'est donc également envisageable que dans un environnement très contrôlé.

– L'hypothèse d'un **bruit Gaussien** dans la formation des images est souvent formulée pour modéliser les erreurs commises lors de l'observation. La couleur du fond observée est supposée soumise à un bruit capteur, dont la distribution est Gaussienne. Les caractéristiques de cette Gaussienne (moyenne et écart-type, ou matrice de corrélation) peuvent être apprises à partir de plusieurs images observées d'une scène fixe par exemple. Plusieurs méthodes connues utilisent cette technique avec quelques variantes [104, 22]. La technique fournit une caractérisation unimodale des pixels du fond (une seule hypothèse pour la couleur de ce pixel). Elle modélise le bruit du capteur, toutefois pour un fond fixe ou comportant peu de changements, car la Gaussienne ne modélise que le bruit capteur et pas les changements pouvant intervenir dans la scène. Cette méthode peut donner lieu à une *carte de probabilité* d'appartenance à la silhouette pour chaque pixel si l'on formule le problème de manière purement probabiliste, résultant de la confrontation d'une nouvelle image avec les caractéristiques du fond. Un seuil peut également être choisi pour garder uniquement les pixels plus pro-

bables et construire une silhouette binaire à partir de la carte de probabilité.

– Pour pallier à certains défauts de la méthode précédente il est possible d'utiliser une **distribution multimodale** pour modéliser plusieurs hypothèses de couleurs possibles pour la couleur d'un même pixel. Le **mélange de Gaussiennes** est classiquement utilisé pour représenter les diverses hypothèses émises et le bruit autour de celles-ci [88, 43, 96]. Ceci permet à l'extraction d'être plus robuste à certains types de mouvements et changements dans la scène : petits mouvements périodiques, variations sous-pixelliques dû à l'échantillonnage pour des pixels se trouvant à la frontière de deux régions de couleurs différentes, etc. Ceux-ci mettent en effet en échec tout modèle unimodal.

– Certaines méthodes utilisent un schéma de prédiction prenant en compte les observations du passé d'un même pixel. C'est le cas de la méthode de Toyama *et al.* [99] qui utilise la prédiction linéaire. Toute couleur observée s'éloignant sensiblement de la prédiction est considérée comme appartenant à l'avant-plan, et donc provenir d'un objet d'intérêt. En pratique la prédiction linéaire, utilisée seule, donne des résultats peu éloignés de ceux obtenus en modélisant la couleur du fond avec une Gaussienne. D'autres schémas de prédiction et mises à jour de modèles de fond plus élaborés existent.

– Il est aussi possible de travailler avec une information plus variée que la simple couleur. La disparité obtenue grâce à une tête stéréo peut par exemple être utilisée comme critère de séparation supplémentaire [28].
D'autres informations locales peuvent être calculées, comme le gradient (filtre de Sobel) ou le Laplacien. Celles-ci permettent de prendre en compte la présence de bords ou de forts contrastes dans l'image. Seule, cette information est généralement utilisé pour fabriquer des détecteurs de bords et de points d'intérêt. En conjonction avec l'information de couleur, elle peut être utilisée dans le cadre de l'extraction de silhouette [55], en supposant que les deux types d'information apportent des indications complémentaires. En pratique, cela n'est que partiellement vrai. Notre expérience du problème tend à montrer que les algorithmes de soustraction de fond sont le plus souvent en difficulté dans des régions aux couleurs uniformes, très sombres ou très claires, c'est à dire où il y a à la fois très peu d'information de couleur et de contraste. On rencontre très typiquement des configurations où le fond et les objets d'intérêt comportent tous deux des couleurs sombres (zones d'ombre, vêtements noirs, plis des vêtements peu illuminés). A ces endroits le fait d'utiliser la couleur seule ou en conjonction avec une mesure

de contraste locale ne fait que peu de différence.

Ajoutons que les décisions portant sur la couleur du pixel peuvent être prises dans différents espaces de couleurs, de préférence dans ceux permettant une meilleure séparabilité des couleurs perçues. Dans cette optique Wu *et al.* [105] suggèrent de travailler avec des intensités logarithmiques qui permettent une meilleure séparation des composantes d'illumination et de reflectance. Pour des raisons similaires, il est généralement utile d'exprimer les images dans un espace comportant des invariants de couleur, et donc de séparer l'information de couleur perçue de celle de luminosité.

A cet effet certaines méthodes de soustraction de fond construisent leur propre espace de couleurs pour obtenir un meilleur pouvoir discriminant : c'est le cas de la méthode de Horprasert *et al.* [53] qui propose un espace luminosité/chromaticité.

Plusieurs autres espaces de couleur existent, qui sont plus adaptés que la traditionnelle décomposition RGB : parmi d'autres, HSV, c1c2c3, l'espace RGB normalisé r*g*b*, et bien sûr l'espace YUV. Notre expérience pratique tend à montrer qu'il y a peu de différences de sélectivité de la silhouette selon si l'on utilise l'un ou l'autre de ces espaces, pour effectuer une soustraction de fond à base de distances entre couleurs et modèles de bruit présentés. Notre préférence va donc naturellement à l'espace YUV, qui est celui directement fourni par la majorité de caméras numériques.

2.3.2 Approches région

Les méthodes d'estimation par pixel présentent l'avantage de la simplicité. Cependant elles présentent certains inconvénients que beaucoup de méthodes cherchent à corriger avec un post-traitement de l'information. En effet l'absence de prise en compte de corrélation entre la décision de pixels voisins fait souvent apparaître des classifications bruitées, spatialement incohérentes et de petites régions isolées aberrantes ("trous"). Ceci comporte bien sûr des répercussions néfastes sur nos algorithmes de reconstruction en aval. Chaque trou dans une silhouette provoque un trou correspondant dans le volume reconstruit à partir des silhouettes. Un bruit trop important dans la détection de silhouette peut engendrer un nombre de calculs important pour un algorithme de reconstruction à partir de silhouettes, s'il traite exhaustivement l'information de silhouette, y compris le bruit ; mais ce bruit est incohérent par nature d'une vue à une autre et donc ne lui apportera aucune information utile.

Un certain nombre de méthodes d'extraction de silhouettes tente de combler ces déficiences avec des heuristiques de remplissage de région dans les images, en propageant des critères locaux, souvent basés sur la présence d'un fort gradient aux frontières, et l'absence de fort gradient au sein d'un région identifiée [99, 55]. La robustesse de telles heuristiques est questionnable s'il ne s'agit que de

propager localement de l'information sur le gradient (par opposition à un algorithme englobant de tels critères dans une fonction de coût globale à minimiser), et peut permettre de propager une erreur locale à toute une région, aggravant potentiellement les erreurs dans certains cas.

Pour pallier à ce type de défauts locaux, il est possible d'utiliser un schéma d'optimisation pour minimiser une fonction de coût sur une région, faisant intervenir la différence de couleur, le gradient, et d'autres critères locaux (texture, disparité dans un voisinage). Les coupures optimales de graphe permettent d'exprimer de tels critères et engendrent des solutions très intéressantes, même si elles ne sont pas toujours entièrement automatiques [87].

2.3.3 Approches contour

Pour formaliser un peu mieux l'idée que les contours de la silhouette doivent coïncider avec des zones de fort gradient dans les images, d'autres méthodes s'intéressent davantage aux contours eux-mêmes qu'aux régions qu'ils renferment. Elles consistent en général à formuler le problème d'extraction de silhouette comme un problème de minimisation d'une énergie calculée sur ce contour. Celle-ci peut prendre en compte la présence de pics de gradient dans l'image, mais aussi un coût lié à l'aspect des régions intérieures et extérieures de la silhouette, se basant sur l'information de couleur ou de texture localement disponible à proximité du contour. Cette vision du problème se prête naturellement à des solutions de type variationnelle. Parmi de telles formulations, on trouve beaucoup de méthodes basées sur les contours actifs [59], qui optimisent le placement d'un contour dans l'espace image. D'autres choisissent une représentation de type level-set, en représentant le contour comme une courbe de niveau d'une fonction définie dans le plan image. Ces méthodes présentent l'avantage, par opposition aux contours actifs, de pouvoir gérer des changements de topologie lors de l'optimisation [107]. Le désavantage de beaucoup des approches contours existantes est la nécessité d'une initialisation manuelle pour fournir une solution proche de la solution finale et éviter la convergence des méthodes vers un mauvais minimum local. Les méthodes n'en demeurent pas moins intéressantes car elles possèdent un fort pouvoir d'expression, et peuvent dans certains cas être initialisées en utilisant une autre méthode moins précise pour pallier à ce défaut. Si les contours actifs peuvent être temps réel, en revanche les approches variationnelles sont généralement beaucoup plus coûteuses et réservées au traitement hors-ligne.

2.4 Silhouettes polygonales

Nous avons vu qu'un éventail important de représentations existaient pour extraire la silhouette à partir d'une image couleur, et éventuellement des observa-

tions du fond. Ces méthodes ont un succès et une exigence de calcul disparates, et sont robustes à certaines conditions mais totalement inadaptées pour d'autres. Il est très difficile de concevoir un algorithme robuste à toutes les situations. Selon le problème à résoudre et les contraintes de temps que l'on se fixe, il convient alors de choisir une approche ou famille d'approches parmi celles qui existent, pour obtenir les silhouettes de la manière la plus adaptée.

Nos conditions d'utilisations, que nous présenterons plus en détail au chapitre 5, sont orientées vers un traitement efficace dans les conditions du temps réel, et dans un environnement statique. Nous trouvons alors un bon compromis entre robustesse d'extraction et rapidité dans les méthodes classiques à base d'une ou plusieurs Gaussiennes pour modéliser les couleurs du fond en chaque pixel.

En particulier, nous sommes souvent amenés à manipuler des cartes de probabilités d'appartenance à la silhouette pour ces pixels, et à effectuer un seuillage de celle-ci pour obtenir une silhouette binaire. Cette dernière représentation est en effet très naturelle à manipuler pour le raisonnement géométrique, avec une partition du plan image en deux régions, intérieures et extérieures à la silhouette. Nous verrons au chapitre 3 que cette représentation des silhouettes donne lieu à une formulation géométrique du problème de reconstruction. Dans le cadre des méthodes alors proposées, nous allons nous intéresser à une représentation polygonale des silhouettes, pour délimiter la frontière entre les régions intérieures et extérieures de celle-ci. Il est en effet possible de déduire une telle représentation à partir de silhouettes binaires, et réduire l'information de manière pertinente pour la reconstruction en aval. Matusik *et al.* [73] proposaient pour ce faire d'utiliser l'algorithme de Marching Cubes, mais celui-ci nous semble sous-optimal pour deux raisons : il introduit des erreurs de sous-échantillonnage dans les contours, et produit un nombre élevé de petites arêtes inutiles, quelle que soit la courbure du contour reproduit.

Une première proposition de cette thèse consiste à montrer que des polygones fournissant une information strictement équivalente aux silhouettes binaires existent et peuvent être calculé avec un algorithme, dit de *vectorisation*. Une telle transformation nous sera d'une grande utilité pour construire des algorithmes efficaces de reconstruction à partir de silhouettes. De plus celle-ci produit des segments se trouvant à une distance bornée des pixels de la silhouette binaire, donnée par la taille du pixel lui-même. La vectorisation décrite n'introduit donc pas d'erreur significative dans la chaîne de traitement.

2.4.1 Principe

L'algorithme présenté s'appuie sur le constat qu'il est possible de reconnaître, à partir d'une séquence de pixels chaînés avec une connexité 8, des segments équivalents au sens discret à ces pixels [29]. Il s'agit d'un processus de reconnaissance réversible, qui est le dual du rendu d'un segment par l'algorithme de

Bresenham.

En effet il est possible de dire, à un ensemble $\mathcal{D}$ de points à coordonnées entières donné, si ceux-ci se trouvent sur une droite discrète de connexité 8. Les points de $\mathcal{D}$ vérifient alors la propriété suivante :

$$\exists(a, b, \mu) \in \mathbb{Z}^3 \mid \forall(x, y) \in \mathcal{D} \qquad \mu \leq ax - by < \mu + \omega \tag{2.1}$$

avec $\omega = \max(|a|, |b|)$. Debled *et al.* [29] fournissent un algorithme de complexité linéaire pour analyser une séquence chaînée de coordonnées et trouver à partir de quel couple de coordonnées la propriété (2.1) n'est plus vérifiée. Ceci permet d'identifier, de proche en proche, les couples de coordonnées correspondant à des sommets qu'il faut conserver pour décrire la séquence à l'aide de segments.

2.4.2 Algorithme de vectorisation des contours

La tâche de vectorisation des contours d'une silhouette binaire peut s'appuyer sur la reconnaissance de segments. Des précautions particulières sont tout de même nécessaires pour garantir que l'ensemble des contours engendrés soit dépourvu d'auto-intersections. Ceci est bien sûr indispensable pour le succès des reconstructions s'appuyant sur ces contours. Nous évoquons donc le processus un peu plus en détail.

L'algorithme de polygonalisation de contour que nous proposons se décompose en quelques étapes, illustrées en figure 2.1. Il repose sur l'identification de pixels "limitrophes" de la silhouette, se trouvant à la frontière entre intérieur et extérieur, et sur la construction d'un contour primitif englobant qui garantit que tous les pixels de la silhouette se trouvent strictement à l'intérieur de ce contour, localement à gauche de chaque arête. Il est en effet très peu pratique de chaîner directement les coordonnées du centre des pixels limitrophes, ce qui engendre de multiples cas particuliers complexes à traiter (portions du polygone avec une épaisseur nulle, auto-intersections dans le contour). Une telle situation se produirait sur les pixels 5,6,7 de la figure 2.1, une "oreille" du contour faisant moins d'un pixel d'épaisseur. La propriété englobante du contour fournit permet de reconnaître un contour polygonal sans auto-intersection.

En revanche les contours ainsi crées n'englobent pas les pixels limitrophes. Ceci n'est pas gênant mais nécessite de traiter certains cas particuliers, comme celui apparaissant entre les pixels 9 et 10 de la figure 2.1. A cet endroit en effet la reconnaissance du contour intérieur peut engendrer un sommet dégénéré, commun entre deux contours. De plus, certaines configurations de pixels très particulières peuvent engendrer des contours contour à deux sommets et donc non génériques. Une détection et un traitement explicite est nécessaire pour éviter ces deux cas dégénérés.

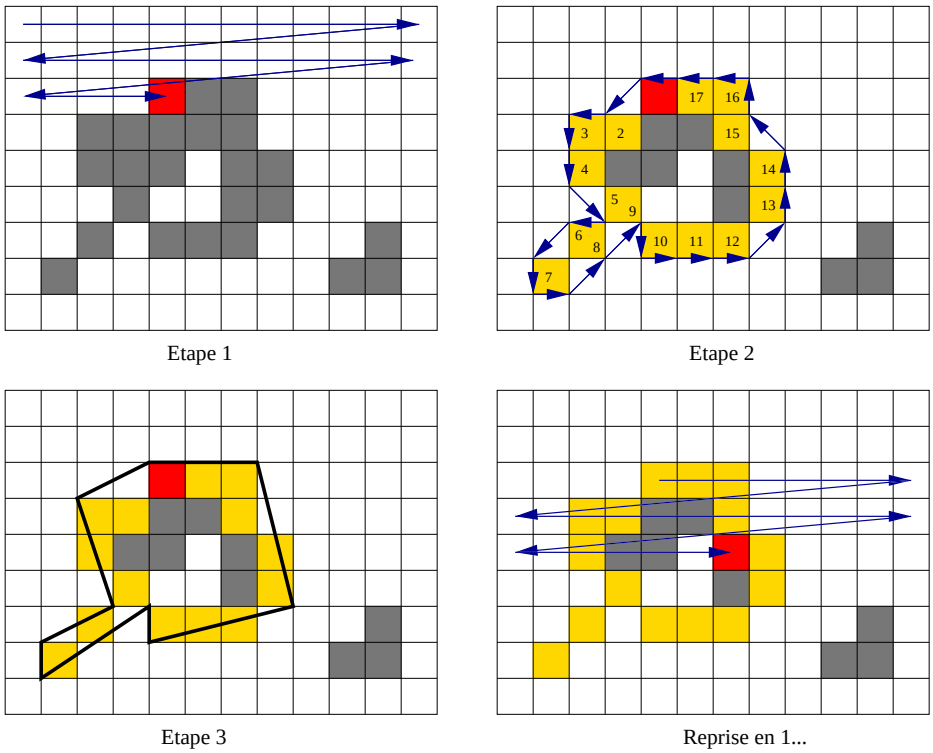


FIG. 2.1 – Détourage et polygonalisation des contours de l'image. Étape 1 : l'image est parcourue dans l'ordre des lignes de balayage pour trouver un point de la silhouette. Étape 2 : le contour de la silhouette est parcouru dans le sens direct en marquant et en pivotant autour des pixels limitrophes, jusqu'à revenir au premier. Un contour strictement englobant est construit de proche en proche à l'échelle du pixel. Étape 3 : on applique [29] à la liste des coordonnées du contour englobant. Étape 4 : parcours dans l'ordre du balayage, recherche d'un pixel limitrophe non déjà marqué pour l'identification d'un nouveau contour à l'étape 2.

2.4.3 Résultats

Nous avons pu implémenter cette solution de vectorisation et avons eu des résultats satisfaisants, illustrés en figure 2.2. Le traitement proposé est effectué en temps réel, et ne prend que quelques millisecondes par image.

La vectorisation de silhouette binaire proposée reconnaît donc aussi bien les contours intérieurs qu'extérieurs et en fournit les segments dans un ordre toujours cohérent : localement, la silhouette est toujours à gauche d'un segment du contour. Une partie du bruit non filtré au cours de la soustraction de fond peut en outre être éliminé : on dispose de la longueur en pixels du contour en cours de traitement, ce qui permet d'éliminer les contours dont la longueur est en dessous d'un certain seuil, et évite par la même occasion certains cas dégénérés. L'algorithme présente le léger inconvénient d'engendrer beaucoup de segments, en effet le bruit dans les images et la soustraction de fond est un obstacle majeur à la bonne restitution des parties rectilignes des contours.

Il est possible d'utiliser certaines stratégies de simplification de contours. En particulier un autre algorithme proposé par Debled *et al.* [30] capable de reconnaître linéairement des segments à un ordre d'erreur près supérieur à la taille du pixel. Cependant, la reconnaissance de contours à partir de tels algorithmes ne se fait pas sans un coût supplémentaire important. Ce coût n'est pas dû à la reconnaissance de segments, mais à la nature des anomalies topologiques à gérer au cours de la simplification et la complexité de leur détection : toute simplification ou construction de contour doit se faire sans anomalie topologique, dégénérescence ou auto-intersection. L'opération est néanmoins rentable, car elle permet de réduire la complexité des entrées fournies aux algorithmes de reconstruction et donc le temps de calcul de telles méthodes.

2.5 Discussion

Nous avons donc effectué dans ce chapitre un tour d'horizon des méthodes et représentations existantes pour extraire une silhouette dans une image. Nous proposons l'utilisation d'algorithmes de soustraction de fond simples et susceptibles d'être temps réel dans notre contexte. Les modèles de fond que nous utilisons sont donc essentiellement des modèles statistiques d'apprentissage des caractéristique de bruit de mesure sur fond fixe, mais d'autres possibilités restent ouvertes en fonction du problème à résoudre. De tels modèles peuvent engendrer une carte de probabilité d'appartenance à la silhouette. Nous proposons également un algorithme de vectorisations de silhouettes pour obtenir des contours polygonaux à partir de silhouettes binaires, obtenues par seuillage d'une carte de probabilité dans notre contexte.

Dans les chapitres à venir, nous allons montrer comment utiliser l'information

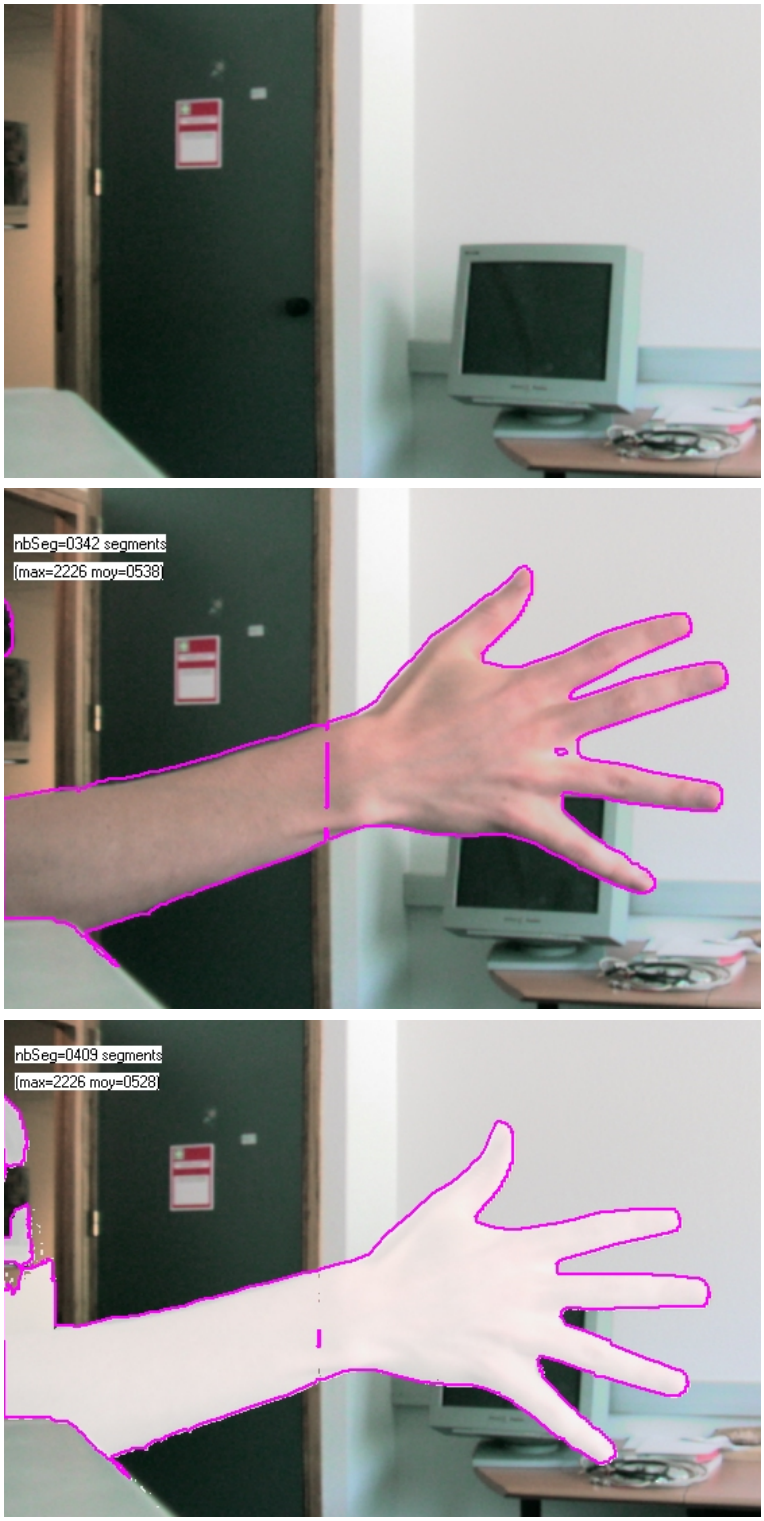


FIG. 2.2 – Polygonalisation d'une main en temps réel. De bas en haut : (1) fond fixe. (2) main segmentée et contour de la main polygonalisé avec notre algorithme. (3) main segmentée et contour polygonalisé ; la silhouette détectée par la passe de segmentation est coloriée en blanc : c'est elle qui est la donnée d'entrée de l'algorithme de vectorisation proposé.

silhouette sous ces deux formes, aussi bien purement géométrique (chapitre 3) que probabiliste (chapitre 4).

3.

Reconstruction géométrique

Nous avons vu, au chapitre précédent, que plusieurs moyens existent pour extraire les silhouettes dans les images. La représentation binaire des silhouettes est sans doute la plus intuitive : elle permet de définir deux régions de l'espace, la silhouette et son complémentaire dans le plan image, séparées par un ensemble de contours. C'est pourquoi le problème de modélisation à partir de silhouettes a tout d'abord été abordé de manière purement géométrique. Reconstruire une forme à partir de silhouettes se définit alors comme le fait de trouver une région de l'espace dont la forme permet d'engendrer les silhouettes observées dans les images.

Il s'agit bien sûr d'un problème mal posé : plusieurs telles formes existent, et sans information supplémentaire, il est impossible de privilégier l'une ou l'autre des solutions possibles. La communauté de vision s'est cependant penchée sur une solution particulière intéressante de ce problème : l'*enveloppe visuelle* [11]. En effet cette solution contient toutes les autres, y compris la forme réelle de l'objet considéré.

L'enveloppe visuelle a suscité beaucoup d'intérêt : elle est une clé intéressante pour trouver des modèles 3D d'objets à partir d'images, dès que l'on dispose de silhouettes fiables de ceux-ci. Il existe dans la littérature plusieurs études de l'enveloppe visuelle [64, 66]. Si celle-ci possède une topologie et une géométrie bien définie, avec une surface délimitante unique, la plupart des travaux existants en construisent des représentations numériques souffrant d'incomplétude, ou laissent de côté les problèmes de topologie et de fermeture de la surface reconstruite. Nous nous penchons plus précisément sur ces questions dans ce chapitre, et proposons deux nouvelles solutions au problème du calcul de la surface de l'enveloppe visuelle, fournissant des surfaces non dégénérées. Il sera aussi question d'identifier les conditions permettant de fournir de telles surfaces, dans un contexte de reconstruction 3D en temps réel, où l'on tente de faire coïncider deux objectifs, souvent antinomiques pour les algorithmes de géométrie : robustesse et rapidité de calcul.

3.1 Introduction

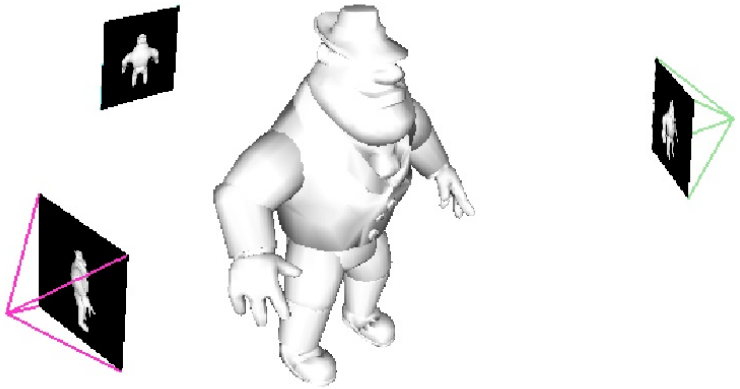


FIG. 3.1 – Trois caméras observent une scène.

Considérons N caméras connues qui observent une scène. On suppose disposer de silhouettes d'un même objet correspondant aux points de vue de ces différentes caméras (voir figure 3.1). L'*enveloppe visuelle* se définit généralement comme la forme maximale cohérente avec les silhouettes de l'objet. L'intuition géométrique du problème de reconstruction à partir de silhouettes a été donnée très tôt par Baumgart, dès 1974, qui proposait déjà une approche à base d'intersection de cônes polyédriques (les "*cônes visuels*" que nous définirons plus loin). La définition de l'enveloppe visuelle telle que nous la connaissons aujourd'hui a été introduite par Laurentini [64] dans un contexte théorique où un nombre infini de points de vue entourant l'objet sont considérés. Il est aussi un des premiers a rendre compte des propriétés fondamentales de cette surface, qui contient la surface réelle de l'objet reconstruit. Avant et après cette contribution, l'enveloppe visuelle a été très étudiée, de manière implicite et explicite, dans les communautés de vision et de graphisme. En particulier, il a récemment été montré [66] que l'enveloppe visuelle d'un objet de surface courbe est un polyèdre généralisé, aux facettes et arêtes courbes, que l'on peut déterminer avec un calibrage faible en utilisant la géométrie épipolaire orientée [65], c'est-à-dire sans connaissance explicite des paramètres de la caméra, mais en conservant la connaissance de l'information d'orientation relative dans les images. Cependant la solution fournie par cet article est peu adaptée au traitement rapide et robuste : les approches que nous proposons ici étendent ces travaux et proposent de nouveaux algorithmes, plus simples et plus fiables. Il existe beaucoup d'autres algorithmes fournissant des approximations de l'enveloppe visuelle dans les deux communautés. Certains s'intéressent au volume délimité par l'enveloppe visuelle et se basent sur des discrétisations de l'espace (approches volumiques). D'autres visent à reconstruire la surface de l'enveloppe visuelle en fournissant des points isolés ou un maillage (approches surfaciques).

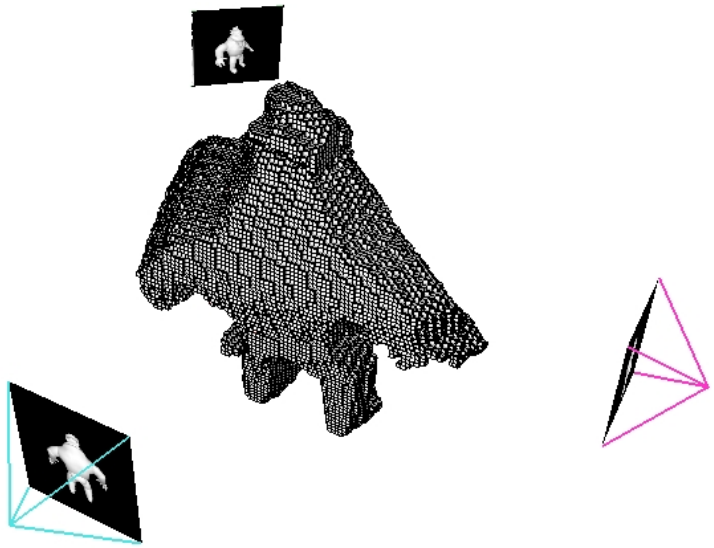
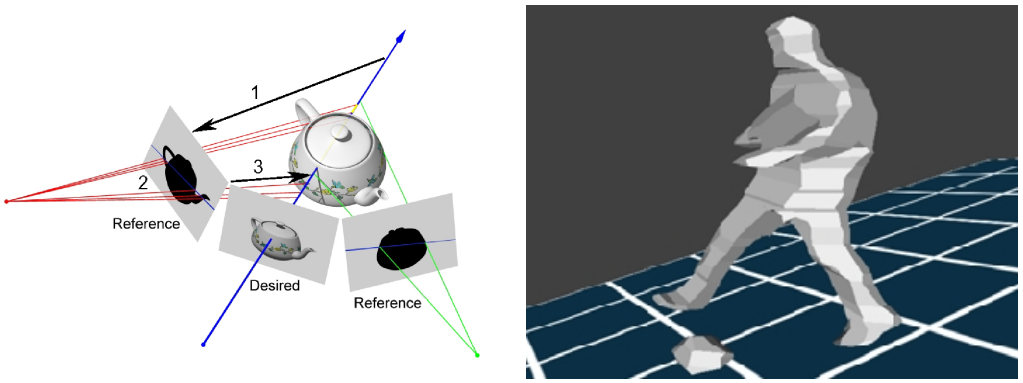


FIG. 3.2 – Reconstructions à partir de primitives volumiques discrètes : les voxels.

Notons qu'une troisième approche existe, calculant une représentation de l'enveloppe visuelle à la résolution des images [74]. Un rendu de l'objet peut en effet être obtenu pour un point de vue quelconque, en calculant pour chaque pixel l'intersection de sa ligne de vue avec l'enveloppe visuelle associée aux silhouettes de référence (voir figure 3.3). Dans les paragraphes suivants, nous précisons les travaux existants pour les deux catégories d'approches les plus répandues, les approches volumiques et surfaciques.

3.1.1 Approches volumiques ou volumétriques

Les approches volumétriques se basent sur une discrétisation de l'espace en cellules élémentaires, les voxels, qui sont "sculptés" au regard de leur projection sur les images et de l'appartenance ou non de celle-ci aux silhouettes de l'objet. Le type de volume reconstruit avec cette approche est illustré en figure 3.2. Une première approche fut proposée par Martin et Aggarwal [72], qui utilisaient des cellules parallélépipédiques alignées sur les axes d'un repère de la scène. Plus tard une représentation adaptative de l'enveloppe visuelle fut proposée [24] sous forme d'octree, mais dans un contexte simpliste (3 caméras affines dans 3 directions orthogonales de l'espace). Ces travaux seront étendus par Potmesil *et al.* [85] puis Srivastava [95] au cas de caméras perspectives. Au cours des années 90, des approches efficaces [98, 83, 22] ont été présentées pour calculer des représentations volumiques. En particulier Szeliski *et al.* proposent un calcul itératif optimisé sur un octree [98], pour concentrer le calcul sur les parties contenant la surface de



(a) (b)

FIG. 3.3 – (a) Image-based visual hulls, tiré de [74]. (b) Polyhedral Visual Hulls for Real-Time Rendering, tiré de [73].

l'enveloppe visuelle, cette fois-ci en gérant un nombre arbitraire de caméras avec un modèle projectif. Niem *et al.* calcule l'occupation d'une grille de voxels en considérant les intersections de l'enveloppe visuelle avec une colonne de voxels [83], et gagne ainsi un ordre de grandeur dans la complexité du calcul de la grille. Cheung, Kanade *et al.* [22] proposent un schéma d'échantillonnage plus robuste aux aléas dans les silhouettes qui fait contribuer pour un voxel donné plusieurs pixels, dont ils précalculent la position dans les images.

Les approches mentionnées sont purement géométriques et ne considèrent pas l'information photométrique. D'autres méthodes [62, 90] l'utilisent en revanche pour sculpter des voxels selon la *cohérence photométrique* de leur projection sur les différentes images. [93, 31] proposent un tour d'horizon détaillé des approches volumétriques. Toutes les approches mentionnées se basent sur une grille régulière de voxels et peuvent traiter des objets de géométrie complexe. Cependant ces approches présentent un rapport désavantageux entre coût des ressources de calcul et précision. Une précision arbitraire peut être atteinte, mais au prix nécessaire d'une augmentation de la résolution de la discrétisation. D'autre part une telle décomposition de l'espace privilégie certaines directions, notamment celles des axes du repère considéré pour la reconstruction, introduisant un certain biais dans la représentation. Leur manipulation n'est en outre pas toujours bien adaptée pour le rendu ou la modélisation de forme. D'où l'intérêt d'une seconde famille d'approches représentant la surface délimitante de l'enveloppe visuelle.

3.1.2 Approches surfaciques

Les approches surfaciques se concentrent sur le calcul d'une représentation explicite de la surface de l'enveloppe visuelle. Des éléments de la surface de l'enveloppe visuelle, tels que des points ou des facettes, sont estimés par intersection des

cônes de vue associés au contours occultants. Considérablement en avance sur son temps, Baumgart [11] propose la première approche de ce type, une reconstruction de forme polyédrique basée sur une approximation polygonale des contours occultants, dès 1974 (voir figure 3.5).

[61, 44, 26, 101, 15] s'intéressent à des points isolés reconstruits en utilisant des approximations locales du second ordre de la surface. Ce faisant, ils émettent une hypothèse supplémentaire, en considérant que la surface engendrant les silhouettes dans les images est lisse et deux fois dérivable. Les conditions supplémentaires de dérivabilité requises limitent la fiabilité de représentation d'objets comportant des arêtes et discontinuités. Elles rendent également difficile le traitement de la fermeture de telles surfaces, notamment dans les situations dégénérées qui apparaissent nécessairement aux endroits de cotangeance entre cônes de vue. Ces problèmes n'ont d'ailleurs pas été traités dans ces travaux. Une exception existe sous la forme d'une solution consistant à calculer des courbes sur la surface de l'objet avec des approximations locales du second ordre, puis d'en effectuer un ré-échantillonnage par tranches selon un axe d'un repère de la scène [70]. La surface reconstruite est non dégénérée, mais le ré-échantillonnage réintroduit des erreurs et omissions potentielles, tributaires de l'axe et du pas d'échantillonnage choisis. D'autres approches reconstruisent des fragments de surface [97, 18], ou des bandes [73] de l'enveloppe visuelle.

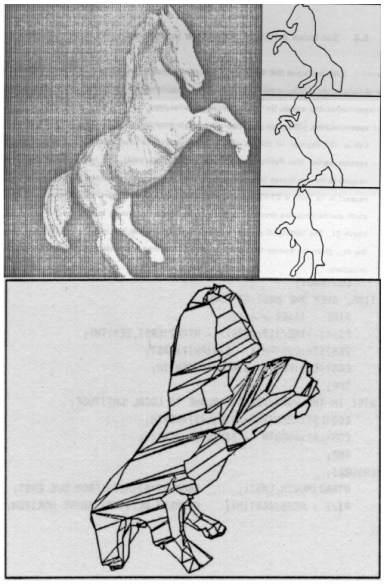


FIG. 3.4 – Modélisation à partir de silhouettes (travaux de thèse de Bruce Baumgart, tiré de [11]).

Les approches surfaciques peuvent être précises comparées aux approches volumétriques, cependant les modèles produits sont souvent incomplets ou erronés, en particulier si l'on considère des objets complexes. Ces anomalies découlent de la sensibilité aux instabilités numériques de ces algorithmes et des calculs impliquant les cônes de vue, dont le lieu d'intersection est mal défini près des points de la surface appelés *points frontière*, lieu de cotangeance de plusieurs cônes. Matusik *et al.* [73] ont montré que l'on peut obtenir une représentation de l'enveloppe visuelle d'un objet en n'effectuant que des calculs 2D. Ce résultat intéressant découle de la structure projective de l'enveloppe visuelle précédemment soulignée par Lazebnik *et al.* [66], et permet de calculer une bande de la surface de l'objet par image. Cependant la méthode proposée ne fournit pas de solution fiable pour connecter ces bandes entre elles. L'auteur mentionne la possibilité d'utiliser des règles de

proximité pour identifier les points communs entre bandes : malheureusement une telle heuristique n'a aucune chance de fournir avec certitude un modèle 3D fermé, sans auto-intersections, comme le requièrent de nombreuses applications. Il est à noter que ce n'est pas ce qui intéresse les auteurs : ils se placent plutôt dans l'optique d'une approche graphique, capable de fournir le strict nécessaire pour effectuer un rendu de l'enveloppe visuelle (les bandes sont dans ce cas suffisantes), sans s'intéresser davantage aux problèmes topologiques de la surface construite. Cette approche a d'ailleurs fait des émules, puisque le concept peut être étendu pour être réalisé sur carte graphique [67, 69]. Cependant les problèmes potentiels de stabilité de calcul au voisinage des zones de cotangeance entre surface, et les moyens permettant d'étendre ce type d'approche pour obtenir une surface topologiquement correcte (fermée et sans auto-intersections) dans un contexte plus exigeant de modélisation, ne sont pas abordés. Il est d'autant plus intéressant de se pencher sur la question que nous montrerons au cours de ce chapitre qu'il est possible d'obtenir de telles surfaces, à moindre coût algorithmique, et en temps réel.

3.1.3 Difficultés

Nous avons vu précédemment que des silhouettes polygonales dans les images induisaient une représentation polyédrique de l'enveloppe visuelle dans l'espace, par intersection des cônes visuels correspondants à base polygonale. Bien que plusieurs représentations aient été proposées dans les précédents travaux énumérés ci-dessus, peu font appel à la représentation naturelle pour représenter la surface d'un tel polyèdre, c'est à dire un maillage triangulé fermé et sans auto-intersections. Cette représentation est aussi la plus utile à des fins de modélisation 3D, de rendu d'objets reconstruits par image de synthèse, et pour calculer des collisions avec d'autres modèles, dans le cadre d'une application interactive, par exemple. Le fait de manipuler un maillage fermé est aussi un prérequis pour tout post-traitement de la surface (lissage, simplification de maillage). Les difficultés inhérentes à la manipulation de tels maillages sont peut-être à l'origine des réticences rencontrées. Pour reconstituer un maillage propre, il faut en effet :

- maintenir des relations d'adjacence rigoureuses entre trois types de primitives : les points du maillage, les segments reliant deux points, et les primitives 2D planes de surface, qui peuvent être des triangles ou des polygones plus généraux.
- définir et respecter une orientation de la surface, de sorte que les primitives définies permettent non seulement de délimiter l'espace en deux régions (ce qui est à l'intérieur et à l'extérieur de l'enveloppe visuelle), mais qu'il soit possible de retrouver localement, à tout endroit de la surface, où est

l'extérieur et l'intérieur, et donc la normale de cette surface.
– s'assurer que le maillage calculé est bien la représentation d'un polyèdre, à savoir que sa topologie est celle d'une variété, qu'il ne présente aucune auto-intersection, que sa surface est fermée, et que toute primitive n'est représentée qu'une et une seule fois dans le maillage.

Ces propriétés sont définies plus formellement dans les ouvrages de référence de géométrie algorithmique tels que [32], et correspondent à la définition d'un *complexe simplicial fermé*. Nous parlerons informellement de *maillage bien formé* dans le reste du document, pour faire référence à un maillage possédant l'ensemble de ces propriétés attendues.

A ces difficultés viennent s'ajouter celles de l'application que l'on vise. Comme mentionné au paragraphe précédant, la spécificité du calcul d'intersection de cônes cotangents induit des problèmes numériques eux aussi spécifiques. De plus, la reconstruction 3D temps réel nécessite des algorithmes rapides. Le fait d'allier robustesse de l'algorithme et rapidité est une difficulté en soi, car la robustesse des algorithmes géométriques se gagne souvent au prix de processus de vérifications lourds, ou de calculs de prédicats à une précision arbitraire, comme nous l'évoquerons en détail un peu plus loin. Les algorithmes existants ont souvent privilégié l'un au détriment de l'autre : en vision et image de synthèse les algorithmes de reconstruction de surface visent généralement la rapidité ou le résultat partiel tout en laissant de côté l'aspect de robustesse ; en géométrie algorithmique les gens s'intéressent prioritairement à des algorithmes entièrement fiables, avant la rapidité.

3.1.4 Lien avec la géométrie algorithmique et l'image de synthèse

Le type de difficultés rencontrées et énoncées au paragraphe précédent nous pousse naturellement à faire le lien avec un domaine de recherche voisin, celui de la géométrie algorithmique. En effet les membres de cette communauté se sont intéressés à plusieurs problèmes connexes, touchant aussi bien à la représentation des primitives 3D mises en jeu, qu'aux algorithmes permettant de les calculer. Très vite (dès les années 80) ils ont identifié les problèmes géométriques importants, sous-jacents à ce type de calcul.

Ces problèmes ont toujours intéressé la communauté graphique, celle de géométrie algorithmique, et celle de vision par ordinateur. Il est donc peu étonnant qu'une thèse comme celle de Baumgart [11] soit mentionnée dans ces trois communautés. Il est l'un des premiers à s'intéresser à la représentation de polyèdres, et aux thématiques connexes : acquisition d'un tel modèle à partir de silhouettes,

algorithmes de rendu de tels modèles pour le graphisme. Il proposa une structure de données pour représenter un maillage polyédrique, les *winged edges*[12], ainsi qu'un ensemble de fonctions (les opérateurs d'Euler) permettant d'ajouter, de retirer, et de modifier les primitives d'un polyèdre tout en garantissant le respect des invariants d'Euler : en particulier la formule $F + S = A + 2 - 2G$, où F est le nombre de faces, A le nombre d'arêtes, S le nombre de sommets, et G le genre du polyèdre, ou nombre de "trous". Il propose également un algorithme d'intersection de polyèdres s'appuyant sur cette structure, qu'il applique à la modélisation à partir de silhouettes, dont il intersecte les cônes visuels. Ce travail est donc, de toute évidence, une source d'inspiration primordiale pour cette thèse.

Si Baumgart pose les bases de notre travail, il manque en revanche à sa contribution une analyse plus poussée de l'algorithme qu'il propose, aussi bien en terme de robustesse numérique qu'en terme de complexité. Ces carences ont été progressivement identifiées, et des solutions diverses proposées, au cours des années 80 et 90, par des travaux fondateurs, aussi bien de la communauté de géométrie algorithmique que de celle d'image de synthèse.

Les chercheurs se sont surtout intéressés au problème plus général de modélisation de solides par opérations booléennes, dont notre intersection est un cas particulier. Ils se sont rendu compte que l'on pouvait définir implicitement un solide en tant que suite d'opérations ensemblistes sur des solides plus simples. Cette famille d'approches est plus connue sous le nom de géométrie constructive sur les solides, ou CSG, que l'on doit à Requicha & Voelcker [86]. Les CSG se fondent sur des primitives solides simples dont le seul prérequis est que l'on sache les délimiter dans l'espace. Aux CSG sont donc associés toute un série de représentations délimitantes, ou *b-rep* (boundary representation), dont le maillage polyédrique est un cas particulier.

Rapidement, des travaux se sont intéressés au calcul permettant de passer d'une telle représentation implicite de solides, à la représentation explicite de sa surface délimitante. Le cas des polyèdres fut abordé assez rapidement par Laidlaw, Hughes *et al.* [63]. L'algorithme qu'ils présentent est de nature similaire à celui que Baumgart avait donné 10 ans auparavant, et affiche, comme celui-ci, une complexité en $O(F1 \times F2)$ pour le calcul d'une opération booléenne sur deux polyèdres de $F1$ et $F2$ facettes. Il s'avèrera par la suite que cette complexité est sous-optimale et peut être réduite à une complexité quasi-linéaire avec un tri approprié des primitives dans l'espace, comme le font remarquer Naylor *et al.* [81]. Dans ce papier, les auteurs optent en effet pour une décomposition de polyèdres sous forme d'arbre binaire de partition de l'espace (BSP), pour traiter les opérations booléennes. Un tel arbre défini en ses feuilles un ensemble de cellules convexes, dont l'union correspond à l'objet initial. Les calculs ensemblistes portant sur deux telles cellules sont beaucoup plus faciles et de complexité linéaire [20], d'où une décomposition de l'opération ensembliste sur l'objet en opérations

sur les cellules convexes le constituant.

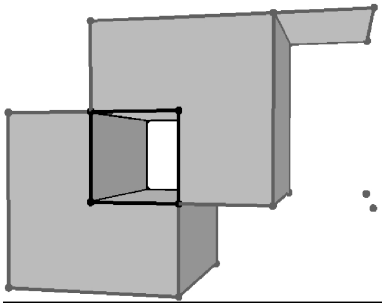


FIG. 3.5 – Polyèdres Nef utilisés par CGAL (tiré de [49]).

Une des approches qui fait référence aujourd'hui en géométrie algorithmique pour calculer des opérations booléennes sur des solides consiste à opter pour une autre forme de partition de l'espace, le polyèdre Nef [82]. Celle-ci repose sur une décomposition de l'espace en demi-espaces ouverts, où les polyèdres sont réalisés par intersection ou complément de ces demi-espaces (ces deux opérations suffisent à modéliser toute opération booléenne). Cette approche est exacte et permet de calculer l'intersection de deux polyèdres avec une complexité $O((F1 + F2 + S)\log(F1 + F2))$, où S est le nombre de primitives supplémentaires créées lors de l'intersection. Elle est implémentée dans la librairie CGAL [2, 49]. Notons au passage que cette complexité est celle à laquelle il faut s'attendre pour tous les algorithmes de type CSG, y compris l'approche originelle de Baumgart, si elle avait été optimisée. En effet les algorithmes d'intersection de b-reps trouvent l'ensemble des facettes qui s'intersectent, puis partitionnent les dites facettes en ajoutant les primitives de surface nécessaires au fur et à mesure de l'itération, de sorte que chaque facette finale soit indivisible et non traversée par une autre b-rep. Ces facette supplémentaires sont alors traitées identiquement aux autres et peuvent être subdivisées à leur tour au cours de l'itération, d'où le terme en S.

Parallèlement s'est posé le problème de la robustesse des calculs géométriques. De tels problèmes ont été soulevés, particulièrement pour le cas des polyèdres. Les difficultés portent principalement sur deux écueils :

– **Précision numérique**. Les opérations booléennes nécessitent de répondre à des questions géométriques, comme celle de savoir si un point se trouve d'un côté ou de l'autre d'un plan, ou encore de trouver l'ordre d'intersections de plans le long d'une droite. Elles nécessitent en outre de calculer des primitives dont les coordonnées sont rationnelles ou homogènes. Se pose alors le problème du pouvoir de représentation des nombres par la machine, dont les types natifs sont de précision fixe et sont sujets à des erreurs numériques d'arrondi susceptibles de rendre de telles décisions imprécises ou incohérentes. La communauté de géométrie algorithmique propose généralement une solution lourde mais exacte : le calcul de *prédicats exacts*, utilisant une précision arbitraire, que l'on doit à Yap [106]. Plus spécifiquement lié à notre problématique, Fortune propose dans cet

ordre d'idée une méthode de calcul d'opérations sur des polyèdres à partir d'arithmétique exacte [35]. Plutôt que d'effectuer tous les calculs avec une précision arbitraire, plusieurs approches proposent d'effectuer des calculs adaptatifs, où la précision n'est utilisée que lorsque l'on détecte des cas ambigus. Le *filtrage flottant* de Fortune & Van Wyk [36] utilise pour ce faire deux représentations pour évaluer des prédicats géométriques : l'une, un type fixe classique de précision limitée, l'autre, un nombre à précision arbitraire fournit par une librairie de multiprécision telle GMP [3]. C'est d'ailleurs cette approche qui est mise en œuvre dans la librairie CGAL, qui se place volontairement dans un contexte de calculs exacts. Une autre approche de représentation intéressante est celle de Shewchuk [92], mettant en lumière la possibilité d'une représentation adaptative des nombres à virgule flottante. Toutes ces solutions, même si elles sont élégantes et édifiantes sur la manière dont la machine peut se représenter et manipuler les nombres, sont très lourdes et pénalisent les algorithmes d'un facteur très important. C'est pourquoi nous cherchons autant que possible à les éviter.

– **Traitement des cas dégénérés**. Pour une opération booléenne sur les solides telle l'intersection, un certain nombre de cas dégénérés peuvent apparaître (coplanarités inattendues, coïncidence mathématique de deux points sur la surface...) qui complexifient considérablement les algorithmes, où 95% du code sert à traiter 5% des cas rencontrés, en pénalisant les 95% de cas génériques en vérifications supplémentaires. Même si l'on veut de la robustesse dans nos algorithmes de reconstruction en vision par ordinateur, c'est typiquement ce que l'on cherche à éviter d'un point de vue efficacité. D'autre part la question de la détection des cas dégénérés est étroitement liée à celle de la précision : en effet ces cas coïncident avec l'évaluation à nulle de déterminants intervenants dans les calculs. Le résultat du test d'égalité que nécessite la détection de tels cas ne peut se faire avec certitude qu'au prix d'une évaluation exacte, à la précision dictée par la forme du déterminant en question (généralement un certain multiple de la précision des données d'entrée). On s'expose sinon à des erreurs d'arrondis et donc à des décisions incohérentes, qui entraîneront une défaillance de l'algorithme. L'impact de telles défaillances a d'ailleurs été analysé en ce qui concerne les calculs ensemblistes sur les polyèdres, notamment par Hoffmann, Hopcroft & Karasick [51]. En conséquence, de nombreux travaux ont été consacrés à la perturbation des entrées de tels algorithmes, pour rendre la formation de tels cas dégénérés défavorable ou impossible, dont une analyse est proposée par Seidel [89]. La perturbation peut être numérique ou symbolique, cette dernière offrant une garantie de résultat, avec l'inconvénient d'un surcoût algorithmique important. Dans le cas où l'on trouve un schéma de pertur-

bation approprié pour le type d'évaluation dont on a besoin, on peut donc se contenter d'écrire l'algorithme qui calcule un résultat uniquement pour les cas non-dégénérés. C'est la stratégie que nous utiliserons.

3.1.5 Contributions

Nous proposons dans ce chapitre deux méthodes pour fournir une représentation maillée polyédrique de l'enveloppe visuelle, dont l'une, préliminaire, en calcule une approximation quasi-exacte, et l'autre calcule l'intersection exacte des cônes visuels. Ces deux méthodes utilisent les spécificités du problème posé à leur avantage : la première en étendant le concept de sculpture d'une grille de voxels à une collection de cellules tétraédriques pour obtenir une surface polyédrique, la deuxième en donnant un algorithme ad hoc capable de traiter en une fois N intersections de cônes, en ne créant et manipulant directement que les arêtes du polyèdre final (algorithme glouton). Les contributions sont donc multiples : par rapport aux travaux existants en vision, nous fournissons deux méthodes nouvelles efficaces pour modéliser le polyèdre de l'enveloppe visuelle à partir des images, avec des garanties supplémentaires de robustesse et de topologie du polyèdre engendré, tout en restant dans le cadre du temps réel. Par rapport aux travaux de géométrie algorithmique, nous donnons des moyens efficaces pour calculer l'intersection de N cônes en une seule passe, plutôt qu'à faire $N-1$ appels à un algorithme générique de type CSG qui calcule des intersections de polyèdres deux à deux. Nous montrons aussi quels choix spécifiques nous pouvons effectuer et quelles contraintes nous pouvons relaxer dans le cadre moins général de notre problème de vision. Ces choix nous permettent de garder l'avantage d'algorithmes simples et rapides à précision fixe, tout en garantissant de calculer une surface bien formée qui réponde au problème de reconstruction 3D posé. Nous calculerons également le coût asymptotique en temps de calcul des approches proposées et les comparerons avec les méthodes de référence.

Nous commencerons par donner quelques définitions utiles pour notre problème. Le contexte général est d'abord présenté, suivi des différentes définitions de l'enveloppe visuelle existantes. Les propriétés de l'enveloppe visuelle, et de sa représentation polyédrique sont ensuite étudiées plus en détail. Les deux méthodes présentées ont en commun de se baser sur un calcul préliminaire de points se trouvant sur la surface de l'enveloppe visuelle : nous détaillerons donc le déroulement de cette étape commune. Puis nous expliquerons le cœur des deux méthodes. Nous ferons une analyse des algorithmes en ce qui concerne la complexité et la robustesse. Enfin, nous donnerons les résultats obtenus par ces méthodes, avec des images synthétiques et réelles. Nous comparerons la méthode polyédrique exacte à d'autres techniques en termes de rapidité et fiabilité. Nous donnerons également, tout au long de ce chapitre, les passerelles nécessaires avec la géométrie algorithmique, domaine de recherche étroitement lié à notre problème et nos objectifs.

3.2 Définitions et notations

Avant de décrire les algorithmes eux-mêmes, commençons par quelques définitions et notations utiles, qui concernent le modèle de caméra utilisé, la géométrie et l'enveloppe visuelle.

3.2.1 Modèles et géométrie

Nous nous plaçons tout au long de cette thèse dans le contexte où N caméras observent une scène, avec des objets d'intérêt. Nous utilisons le modèle classique de caméra à sténopé. Nous présentons plus en détail ce modèle dans l'annexe A, accompagné des notions de géométrie projective permettant de décrire la relation de projection d'un point sur le plan image d'une telle caméra, lorsque ce point est exprimé en coordonnées homogènes.

Nous manipulons un certain nombre de primitives géométriques : points, droites, polygones. Un point de l'espace 3D sera noté X (majuscule) et sera assimilé, par abus de notation, à son vecteur de coordonnées homogènes, à 4 dimensions. Toute entité de l'espace image sera notée en minuscule, en particulier un point image sera souvent noté x ou p, et une droite l. Similairement nous assimilerons de telles entités aux vecteurs de coordonnées homogènes qui les définissent (dimension 3). Nous notons le numéro d'image associé à une primitive en exposant lorsque cela est pertinent.

A toute caméra est associée une *matrice de projection* permettant d'exprimer la relation entre un point X de l'espace et sa projection dans les images x. A chaque point x d'une image est associée sa *ligne de vue*, c'est à dire l'ensemble des points de l'espace se projetant sur le point x. Pour une caméra à sténopé, il s'agit d'une droite passant par le centre de projection et le point image x. Des détails supplémentaires sont donnés en annexe A ainsi que dans les ouvrages de références de géométrie en vision, par exemple [50, 34].

3.2.2 Contours

Considérons une scène composée de plusieurs objets, observée par un ensemble de caméras à sténopé. Nous supposerons la surface des objets fermée et orientable, courbe ou polyédrique. En outre elle peut être de genre non nul. Les *contours d'occultation* (voir figure 3.6) sont, par convention dans cette thèse, le lieu des points sur la surface des objets où les lignes de vue sont strictement tangentes à cette surface. Les contours d'occultation définissent, par projection dans les images, les *contours occultants* [72] qui délimitent la silhouette des objets dans chaque plan image. La topologie de chaque contour occultant, tel que défini ici, est donc celle d'une variété unidimensionnelle. Ces conventions sont légèrement différentes de celles utilisée dans d'autres travaux, où les contours d'occultations sont plus

généralement tangents à la surface et peuvent donc engendrer des contours occultants comportant des jonctions en T. Nous choisissons cette convention voisine, qui permet de faire le lien entre les propriétés topologiques de tels contours occultants et des surfaces que nous allons calculer.

Dans le cadre de ce chapitre, nous notons les numéros de contour en indice. Ainsi, $\mathcal{O}_j^i$ désignera le $j$ème contour occultant dans l'image i. La silhouette qui se forme dans une image peut être de genre non nul : elle peut être composée de plusieurs composantes connexes, chacune pouvant elle-même comporter des trous. C'est pourquoi nous définissons une orientation pour les contours occultants dans les images, de telle façon que la silhouette de l'objet qu'elle délimite est toujours localement à gauche du contour. L'orientation des contours extérieurs et intérieurs de l'objet, respectivement directe et indirecte, découle de cette définition. Nous appellerons *région intérieure* d'un contour occultant la région fermée du plan délimitée par le contour et contenant la silhouette, et nous appellerons *région extérieure* son complément dans le plan image.

3.2.3 Cônes de vue

Nous avons plusieurs fois manipulé informellement la notion de cône de vue dans nos précédentes descriptions. Il s'agit là d'une notion importante pour le calcul de l'enveloppe visuelle : le cône de vue est l'entité géométrique qui définit la contribution d'une silhouette à ce volume. Nous allons distinguer plusieurs définitions : celle du cône de vue *associé à un contour occultant*, puis celle d'un cône de vue *associé à un point de vue*. La nécessité d'une telle distinction vient du fait qu'il peut y avoir, dans le plan image d'un point de vue, plusieurs contours occultants extérieurs correspondants à plusieurs objets de la scène, chacun comportant potentiellement des contours intérieurs. Nous nous consacrons ici à la définition du cône de vue associé à un contour : les deux autres définitions serons introduites en même temps que l'enveloppe visuelle dans le prochain paragraphe, où leur utilité prendra tout son sens.

Intuitivement, le cône de vue associé à un contour occultant est un cône dont le sommet est le centre optique de la caméra et dont la base est la région intérieure de ce contour. Plus formellement le *cône de vue* $\mathcal{V}_j^i$ associé au contour $\mathcal{O}_j^i$ est la fermeture de l'ensemble des rayons passant par les points de la région intérieure de $\mathcal{O}_j^i$ et par le centre de projection de l'image i. Ainsi $\mathcal{V}_j^i$ est un volume tangent à la surface de l'objet qui lui correspond. Le lieu de tangence entre le cône visuel et l'objet est une courbe sur la surface de l'objet, en l'occurrence le contour d'occultation, qui se projette en $\mathcal{O}_j^i$. Selon l'orientation de $\mathcal{O}_j^i$, contour extérieur ou intérieur, le cône de vue $\mathcal{V}_j^i$ tel que défini correspond à un volume projectif dont la base dans le plan image est un ensemble respectivement fermé ou ouvert de $\mathbb{R}^2$.

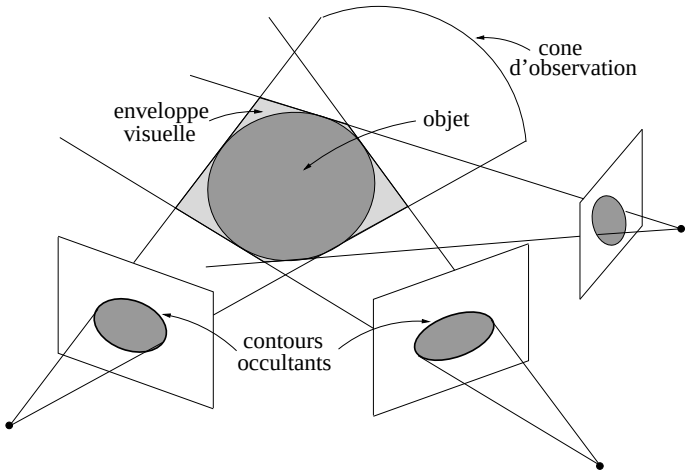


FIG. 3.6 – L'enveloppe visuelle (vue en coupe)

3.2.4 Enveloppe visuelle : définitions ensemblistes

On doit la définition du terme enveloppe visuelle à Laurentini [64], même si la notion fut implicitement manipulée auparavant. L'étude de Laurentini se place dans un contexte théorique, avec une infinité de points de vue pris dans une région fermée de l'espace $\mathbb{R}^3$. Il définit d'ailleurs l'enveloppe visuelle en tant que forme maximale cohérente avec les silhouettes de points vues pris dans une région extérieure à l'objet et entourant celui-ci, mais ne fournit pas de solution pratique pour construire une représentation explicite de la forme de l'enveloppe visuelle. Nous fournissons ici des formulations ensemblistes de l'enveloppe visuelle plus pratiques pour nos applications, dans un contexte réel avec un nombre de points de vues finis. Celles-ci peuvent cependant être facilement étendues pour une infinité de points de vue. Nous nous intéressons aussi à des définitions flexibles et générales, qui prennent en compte le cas où toutes les caméras ne voient pas les mêmes régions de la scène. Beaucoup de méthodes existantes ne s'intéressent en effet qu'au cas où toutes les caméras voient les objets à reconstruire, qui est un cadre plus restrictif que celui proposé.

Informellement, l'*enveloppe visuelle* est définie comme l'intersection de tous les cônes de vue associés aux points de vue considérés. Elle constitue donc une région fermée de l'espace dont les points se projettent à l'intérieur de tous les contours occultants. Soit $\mathcal{I}$ l'ensemble des images considérées, et $\mathcal{C}$ l'ensemble des contours considérés. Dans le cas où la scène ne comporte qu'un seul objet, une telle définition de l'enveloppe visuelle peut être écrite de la manière suivante :

$$\mathcal{VH}(\mathcal{I},\mathcal{C}) = \bigcap_{i\in\mathcal{I},j\in\mathcal{C}^i} \mathcal{V}^i_j,$$

où $\mathcal{V}_j^i$ est le cône de vue associé au contour occultant j dans l'image i. Lorsque l'on considère un ensemble fini $\mathcal{I}$ d'images, la surface de l'enveloppe visuelle est un polyèdre généralisé constitué de fragments de cônes délimités et raccordés par des courbes d'intersection de cônes [66]. En pratique, les contours occultants sont approximés par des courbes polygonales 2D, induisant donc des cônes de vue et une enveloppe visuelle polyédriques. La définition ci-dessus est valide dans le contexte où chaque caméra de $\mathcal{I}$ voit un seul et même objet. Mais elle est incorrecte dans le cas de scènes constituées de plusieurs objets distincts, engendrant plusieurs contours extérieurs dans les images.

Dans ce dernier cas la silhouette formée dans les images est constituée de plusieurs composantes k, chacune étant constituée d'un contour extérieur et éventuellement un ou plusieurs contours intérieurs. Ce constat donne lieu à une définition plus générale de l'enveloppe visuelle, pouvant s'écrire de la manière suivante :

$$\mathcal{VH}(\mathcal{I},\mathcal{C}) = \bigcap_{i\in\mathcal{I}} \Big(\bigcup_{k\in\mathcal{K}^i} \Big(\bigcap_{j\in\mathcal{C}_k^i} \mathcal{V}_j^i \Big) \Big), \tag{3.1}$$

où $\mathcal{K}^i$ est l'ensemble des composantes de la silhouette dans l'image i et $\mathcal{C}_k^i$ est l'ensemble des contours associés à la composante k dans i. Cette expression définit l'enveloppe visuelle en tant qu'intersection de contributions d'une image, chacune de ces contributions étant l'union des contributions de toutes les composantes de la silhouette dans l'image associée. Chaque composante est elle-même l'intersection des différents cônes de vue associés aux contours de la composantes.

Néanmoins, l'expression (3.1) comporte un effet de bord indésirable, puisqu'elle ne prend pas en compte le fait que certains objets peuvent être en dehors du champ de certaines caméras. Avec cette définition, de tels objets ne contribueraient pas à l'enveloppe visuelle (voir figure 3.7-(b)). Cette définition est cependant celle qui est implicitement utilisée par les approches volumétriques pour sculpter les voxels. Les contributions de chaque image ne devraient être prises en compte que dans le domaine de visibilité de cette image. Une astuce consiste alors à calculer le complémentaire de l'enveloppe visuelle dans l'union des domaines de visibilité des caméras. Il s'agit d'une région ouverte de $\mathbb{R}^3$ définie par :

$$\begin{aligned} \mathcal{VH}^c(\mathcal{I},\mathcal{C}) &= \mathcal{D} \setminus \left[\bigcap_{i\in\mathcal{I}} \Big(\bigcup_{k\in\mathcal{K}^i} \Big(\bigcap_{j\in\mathcal{C}_k^i} \mathcal{V}_j^i \Big) \Big) \right], \\ &= \bigcup_{i\in\mathcal{I}} \Big(\bigcap_{k\in\mathcal{K}^i} \Big(\bigcup_{j\in\mathcal{C}_k^i} \mathcal{D} \setminus \mathcal{V}_j^i \Big) \Big) \end{aligned} \tag{3.2}$$

où $\mathcal{D}$ est l'union des domaines de visibilité de toutes les images dans $\mathbb{R}^3$ et où $\mathcal{D}\setminus\mathcal{E}$ est le complémentaire d'un ensemble donné $\mathcal{E}$ dans ce domaine de visibilité. En utilisant (3.2), les objets qui n'apparaissent pas dans une image peuvent encore

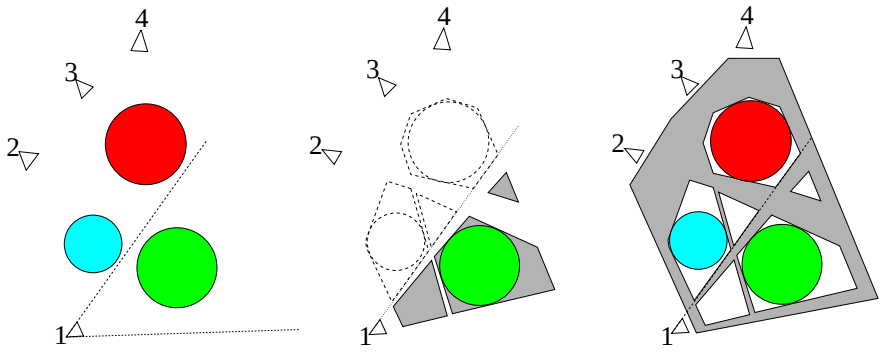


FIG. 3.7 – Une situation à 4 points de vue : impact des différentes définitions de l'enveloppe visuelle.

contribuer à l'enveloppe visuelle puisque les contributions vides ne changent pas la contribution des autres images dans l'expression ci-dessus. La connaissance de l'enveloppe visuelle ou de son complémentaire est équivalente dans la mesure où la surface qui nous intéresse délimite ces deux régions.

L'utilisation des différentes définitions est illustrée en figure 3.7. On y voit une scène à 4 points de vue où la caméra 1 ne voit que l'objet vert, en (b) l'utilisation de l'expression (3.1) : l'enveloppe visuelle (grisée) ne contient aucune contribution relative aux objets rouge et bleu. En (c) avec l'expression (3.2), le complément de l'enveloppe visuelle est calculé, et inclut les contributions des objets rouge et bleu. Notons que l'utilisation de (3.1) et (3.2) induit l'ajout potentiel d'objets virtuels qui n'apparaissent pas dans la scène d'origine (voir figure 3.7). Ces objets virtuels indésirables, ou *objets fantômes*, correspondent à des régions de l'espace qui se projettent à l'intérieur d'une silhouette dans toutes les images. La taille et le nombre de ces objets fantômes peuvent généralement être réduits en augmentant le nombre de caméras.

Nous profitons de ces définitions pour faire le lien avec la notion de *cône visuel associé à une vue i*, noté $\mathcal{V}^i$, qui définit la contribution d'une vue à l'enveloppe visuelle. On peut en effet considérer que l'une et l'autre des définitions (3.1) et (3.2) peuvent se décomposer en N termes propres à une seule vue. Dans le cas de la définition (3.1), le cône $\mathcal{V}^i$ s'exprime de la manière suivante :

$$\mathcal{V}^i = \bigcup_{k \in \mathcal{K}^i} \left(\bigcap_{j \in \mathcal{C}_k^i} \mathcal{V}_j^i \right) \tag{3.3}$$

En ce qui concerne la définition (3.2) de l'enveloppe visuelle, l'expression du cône visuel associé à une vue s'exprime cette fois de la manière suivante :

$$\mathcal{V}^i = \bigcap_{k \in \mathcal{K}^i} \left(\bigcup_{j \in \mathcal{C}_k^i} \mathcal{D} \setminus \mathcal{V}_j^i \right) \tag{3.4}$$

Il est important de garder ces deux définitions à l'esprit lorsque l'on manipule l'enveloppe visuelle. La première, la plus simple et intuitive, permet le calcul d'enveloppe visuelle d'objets visibles de toutes les caméras (3.1) ; la deuxième permet de prendre en compte le cas où les objets d'intérêt ne se trouvent pas dans un domaine de visibilité commun à toutes les caméras, au prix de l'ajout d'objets fantômes supplémentaires (3.2). Nous utiliserons l'une et l'autre de ses définitions dans le reste de cette thèse.

3.3 La surface de l'enveloppe visuelle

Nous avons vu comment définir l'enveloppe visuelle de manière ensembliste comme intersection de cônes visuels. Nous nous intéressons maintenant plus particulièrement aux propriétés associées à une telle définition sur la surface délimitant l'enveloppe visuelle, que nous cherchons à reconstruire. Pour cela nous étudions la structure de cette surface, et en particulier comment celle-ci se présente lorsque l'on opte pour une représentation polygonale des contours occultants. L'enveloppe visuelle prend alors une forme polyédrique que l'on peut calculer avec les deux solutions que nous proposons.

La structure de l'enveloppe visuelle dans le cas d'un nombre fini de points de vue a été l'objet d'études approfondies. Certaines de ses propriétés de l'enveloppe visuelle ont récemment été soulignées par Lazebnik *et al.*, dans un contexte où les silhouettes d'objets sont représentées avec des contours occultants lisses [66]. Ces propriétés sont relativement génériques et permettent de se faire une idée plus précise de ce qu'est l'enveloppe visuelle, c'est pourquoi nous les rappelons ci-dessous. Néanmoins dans le contexte où nous nous plaçons, le calibrage des vues peut comporter une erreur numérique, et nous disposons de silhouettes polygonales, représentation exacte au regard de nos entrées de nature discrète, les pixels. La représentation induite par ce type d'entrées est significativement différente. C'est pourquoi nous consacrons trois autres paragraphes à faire le lien entre ces deux visions de l'enveloppe visuelle. Nous y abordons l'impact du bruit, de la discrétisation des contours sur la représentation de l'enveloppe visuelle, et en détaillons alors la structure dans ce contexte. Les algorithmes que nous introduirons reposent en effet sur les propriétés structurelles identifiées ici.

3.3.1 Cas des silhouettes aux contours lisses

Nous décrivons ici l'étude du cas idéalisé exposé par Lazebnik *et al.* [66], avec un calibrage parfait et des silhouettes aux contours occultants lisses par morceaux. Nous nous intéressons donc ici à la structure l'enveloppe visuelle induite par de telles contours.

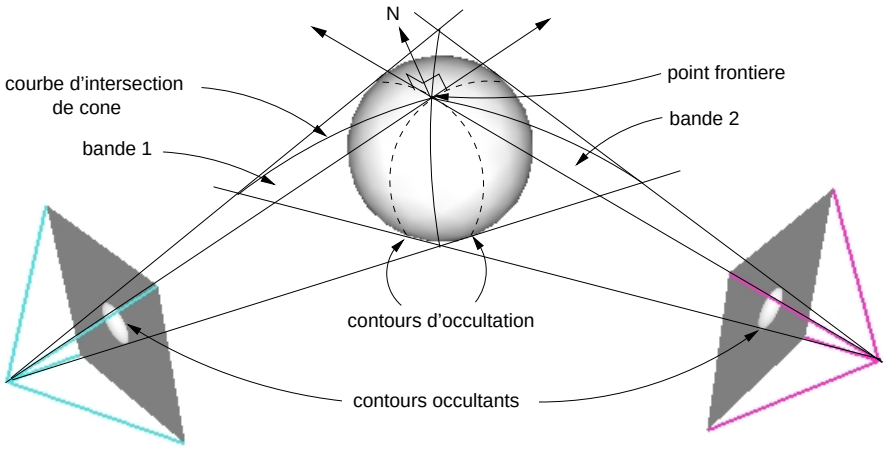


FIG. 3.8 – Structure de l'enveloppe visuelle et contributions de deux vues.

Dans un tel contexte, les primitives de l'enveloppe visuelle sont des surfaces ou des courbes lisses. Les surfaces des cônes de vue $\mathcal{V}_j^i$ s'intersectent dans l'espace en des courbes dont les points n'appartiennent pas à la surface des objets, à l'exception de certains points appelés *points frontière*, lieux d'intersection des contours d'occultation, et de cotangeance des cônes. Entre les courbes d'intersection de cône se forment par intersection des bandes, qui correspondent à des portions de surfaces de cônes tronqués. Ces bandes constituent les primitives surfaciques de l'enveloppe visuelle, et correspondent à la contribution d'une seule vue à celle-ci. Voir la figure 3.8 pour une schématisation des primitives mentionnées.

Notons qu'une bande est facilement orientable, car elle hérite naturellement de l'orientation du contour image duquel elle est issue. Nous appellerons *haut* la direction induite par l'orientation du contour dans les images (voir figure 3.9), et *bas* la direction opposée sur la bande. De telles conventions s'avèreront utile pour orienter la surface durant sa reconstruction.

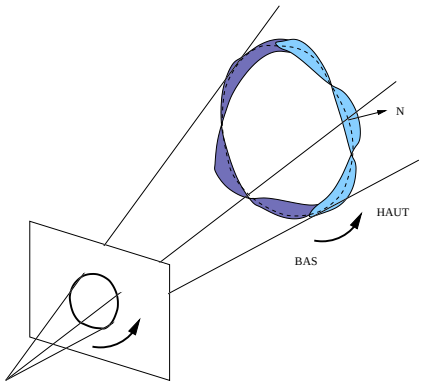


FIG. 3.9 – Orientabilité d'une bande de l'enveloppe visuelle, héritée de l'orientation du contour qui l'a engendrée.

L'enveloppe visuelle comporte également des contributions faisant intervenir trois vues, sous la forme de points sur sa surface appelés *points triples*. Ces points triples correspondants à l'intersection de trois cônes de vue, et apparaissent donc au carrefour des trois courbes d'intersection de cônes, faisant chacune intervenir un couple de vues parmi les trois engendrant le point triple.

Au final, l'ensemble de ces primitives présente une organisation particulière, un maillage de

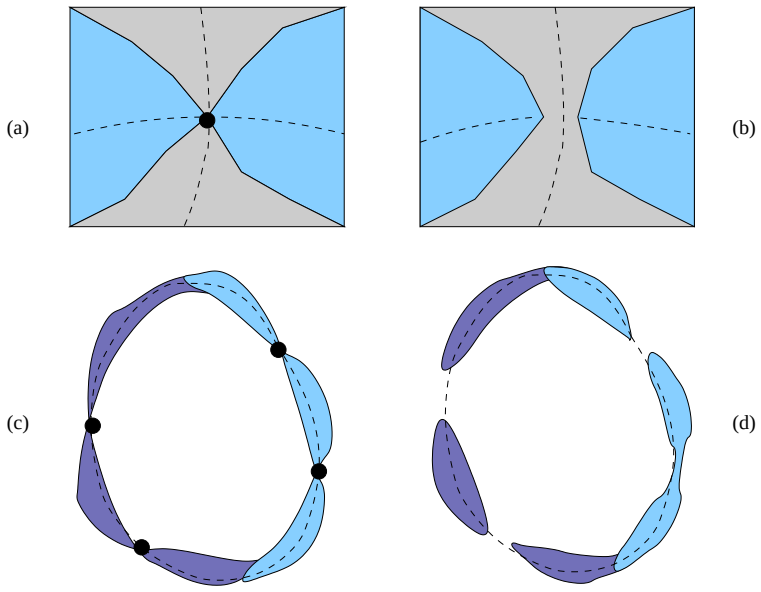


FIG. 3.10 – Enveloppe visuelle schématique obtenues avec des silhouettes lisses : (a) cas idéal : deux bandes et deux contours d'occultation se rencontrent en un point frontière. (b) cas réel : la cotangeance entre cônes est perdue, une bande passe par dessus l'autre. Aspect d'une bande isolée dans le cas idéal (c) et le cas réel (d).

points frontières et points triples, reliés par des intersections de cônes. L'enveloppe visuelle se présente ainsi sous la forme d'un *polyèdre généralisé*, avec des facettes et arêtes courbes, dictées par les contours des silhouettes dans les images. Sa structure est projective, et peut donc être entièrement déterminée avec la seule connaissance d'un calibrage faible de l'ensemble des caméras (géométrie épipolaire orientée).

3.3.2 Cas réel

L'étude du paragraphe précédent montre la structure de l'enveloppe visuelle dans un contexte idéalisé. Cependant les hypothèses de calibrage et contours parfaits sont irréalistes. Le calibrage, obtenu grâce à des méthodes numériques [54], et l'extraction de silhouette, sont tous deux sources d'erreur. En conséquence, la cotangeance entre cônes visuels qui se produit dans le cas idéal est instable en pratique, et les points frontières qui marquent le lieu de ces cotangeances sont donc en général perdus (voir figure 3.10). Cependant l'enveloppe visuelle elle-même, qui est définie en tant qu'intersection volumique de cônes, est stable à ces petites erreurs. C'est pourquoi il est encore possible d'en calculer une représentation malgré ces diverses sources de bruit.

Il existe entre le cas réel et le cas idéalisé une différence fondamentale. Le cas idéalisé considère un ensemble de silhouettes obtenues sans aucune erreur, via un ensemble de caméras dont les paramètres sont parfaitement connus. L'enveloppe visuelle qui résulte de ces entrées est alors naturellement une et unique, et admet un ensemble de propriétés étudiées au paragraphe précédent. Dans le cas réel, les différentes sources de bruit introduisent des erreurs, que nous supposons bornée : nous ne manipulons donc plus une seule enveloppe visuelle, mais une famille de volumes, qui représentent toutes la même enveloppe visuelle idéale, à une erreur métrique près.

Nous introduisons l'a priori d'une erreur bornée : en d'autres termes nous supposons qu'une silhouette est projetée dans les images avec une erreur reprojection maximale de ϵ par rapport à la silhouette idéale qui serait obtenue sans erreur d'observation. Cette hypothèse de travail nous semble réaliste puisque les erreurs introduites par la segmentation restent locales, et que l'erreur de calcul commise sur les paramètres de calibrage obtenus par des méthodes standard est bornée. En conséquence, nous manipulerons dans le reste de ce chapitre non pas l'enveloppe visuelle idéale, mais une *enveloppe visuelle à ϵ près*, c'est à dire un des volumes parmi l'infinité de volumes existants projetant des silhouettes dont l'erreur de reprojection dans les images est au maximum ϵ par rapport aux silhouettes idéales. Le cas échéant nous pourrons en outre choisir, par perturbation numérique, une de ces représentations permettant de construire une surface générique, sans primitive dégénérée sur la surface, comme nous le détaillerons.

3.3.3 Polyèdres équivalents à l'enveloppe visuelle

Nous avons vu quelles étaient les conséquences des imperfections des diverses erreurs de calibrage, et de segmentation, sur la surface de l'enveloppe visuelle. Nous examinons ici l'impact de la discrétisation des silhouettes sur celle-ci.

De toute évidence le fait de disposer de contours polyédriques pour les silhouettes induit une enveloppe visuelle *polyédrique* à ϵ près. Ceci vient du fait que la discrétisation des silhouettes que nous utilisons, présentée au chapitre 2, n'introduit qu'une erreur bornée, de l'ordre de la taille du pixel. Pour calculer un tel polyèdre il suffit d'appliquer les définitions ensemblistes du paragraphe 3.2.4, en calculant par exemple l'intersection de cônes visuels associés aux contours polygonaux dont on dispose.

Nous ferons pour tout ce chapitre une hypothèse forte : l'existence systématique de polyèdres génériques, **aux sommets tous trivalents**, représentant l'enveloppe visuelle à ϵ près. Nous appellerons ces polyèdres *enveloppes visuelles polyédriques* (EVP) dans la suite de cette thèse. L'hypothèse d'existence de tels polyèdres résulte de l'intuition que tout cas de dégénérescence sur la surface d'un polyèdre (intersection de plus de trois plans en un même point de l'espace, plans confondus, cotangences et alignements impromptus) est très

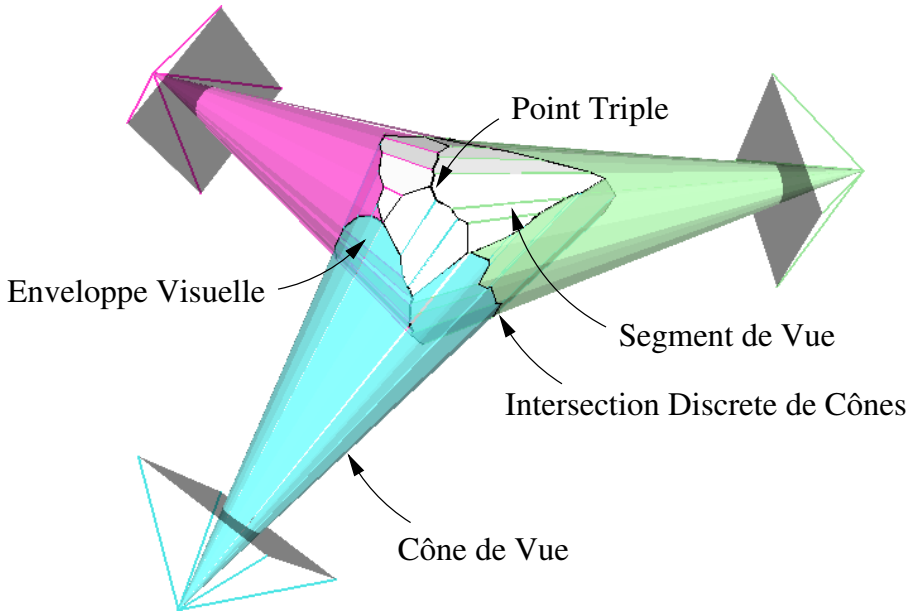


FIG. 3.11 – EVP d'une sphère obtenue à partir de trois silhouettes polygonales, et les primitives associées.

improbable, qu'il soit propre à l'enveloppe visuelle (points frontières) ou non. En effet les erreurs introduites par la segmentation, le calibrage, la discrétisation, et la représentation discrète des nombres par la machine, rendent selon nous l'instantiation numérique de tels dégénérescences très instable et improbable en général.

Nous travaillons néanmoins avec des polyèdres toujours proches d'un état dégénéré. Pour diminuer davantage la probabilité que le traitement des contours donnés en entrée aboutissent à un polyèdre dégénéré, nous appliquons une perturbation arbitraire des sommets des silhouettes polygonales à une échelle très inférieure à ϵ, induisant donc toujours un polyèdre équivalent.

Si nous ne donnons pas de preuve du bon fonctionnement d'une telle manipulation, nous constatons qu'elle nous permet quasiment toujours de calculer un polyèdre générique équivalent à l'enveloppe visuelle en pratique. Nous avons pu développer sur la base de ces hypothèses un algorithme reconstruisant des objets à partir de silhouettes avec un taux de réussite proche de 100%, dans une expérience décrite au paragraphe 3.9.5 faisant intervenir de nombreux modèles. Nous décrirons donc dans la suite de ce chapitre comment modéliser l'enveloppe visuelle d'objets en utilisant une représentation par polyèdres aux sommets trivalents.

3.3.4 Structure d'une enveloppe visuelle polyédrique

Une enveloppe visuelle polyédrique induite par un jeu de contours polygonaux est représentée en figure 3.11. Toute EVP induite par des contours polygonaux est aussi structurée en bandes. Ces bandes ne sont plus lisses mais planes par morceaux, car elles correspondent à des portions de cônes visuels polyédriques tronqués. Chaque bande est fragmentée en facettes ou groupe de facettes coplanaires, correspondant à la contribution d'une arête e des contours dans les images. Chaque arête e engendre en effet une facette triangulaire infinie $\mathcal{T}_e$ du cône visuel correspondant. Le groupe de facettes constituant la contribution de l'arête e se trouve donc naturellement inclus dans $\mathcal{T}_e$.

Les facettes d'une même bande sont incidentes à des arêtes appelées *segments de vue*. Il s'agit d'arêtes se trouvant sur la ligne de vue d'un sommet du polygone représentant un contour occultant dans les images. Les segments de vue constituent la contribution à l'enveloppe visuelle de ce sommet. Comme nous le verrons un peu plus loin il est possible de les calculer efficacement (paragraphe 3.4), ce qui s'avèrera primordial dans le fonctionnement des algorithmes proposés pour le calcul d'EVP.

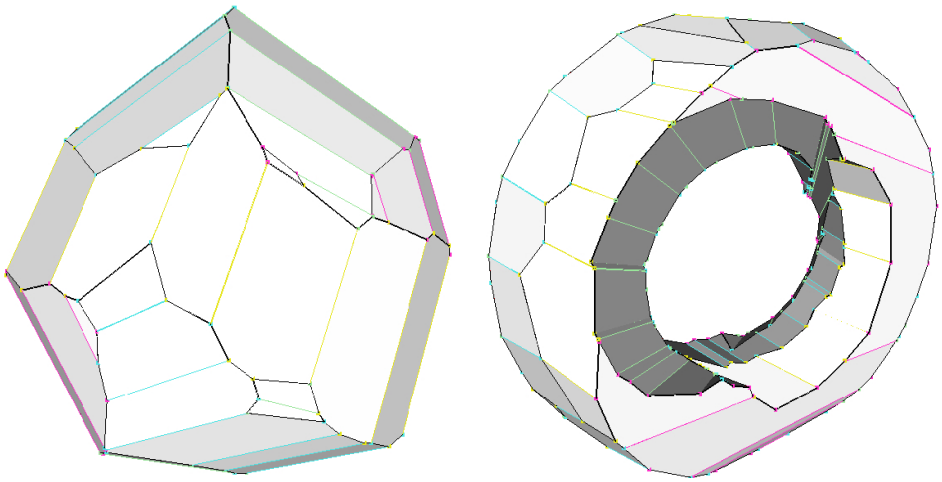
La structure des bandes des EVP est mise en évidence dans les figures 3.11 et 3.12. A noter les ramifications possibles d'une bande en plusieurs fragments ou branches, en conséquence des concavités apparaissant dans les images, engendrant des parties hyperboliques sur la surface, ce qui est illustré par le cas du tore dans la figure 3.12.

Les silhouettes polygonales induisent également une discrétisation des courbes d'intersection de cônes : celles-ci ne sont plus lisses, mais linéaires par morceaux (voir figure 3.11 et 3.12). En revanche elles conservent toujours la propriété de se rencontrer en des points triples. En effet le point triple, lieu d'intersection de trois cônes, reste stable et bien défini, malgré la discrétisation. Il ne résulte pas d'une situation dégénérée sur la surface.

Il est important de garder à l'esprit que toute primitive se trouvant à la surface d'une EVP se projette sur un ou plusieurs contours des silhouettes dans les images et à l'intérieur de toutes les autres silhouettes - ou, de manière équivalente, toute primitive se trouve sur la surface d'un ou plusieurs cônes visuels, et à l'intérieur de tous les autres cônes visuels. Ceci découle du fait que l'EVP se définit comme l'intersection des cônes visuels en question.

En particulier :

1. Une facette du polygone se projette sur un seul contour occultant dans une vue i, et à l'intérieur de toutes les autres silhouettes dans les vues $v \neq i$. C'est aussi vrai des segments de vue.
2. Un sommet de segment de vue se projette sur deux contours occultants dans deux vues différentes i et j, et à l'intérieur des silhouettes dans toute vue



(a) (b)

FIG. 3.12 – (a) EVP d'une sphère obtenue à partir de silhouettes polygonales, pour quatre points de vue. Les segments de vue sont dessinés en couleurs, une par caméra, mettant en évidence sa structure en bandes. Symptômes d'une tangence perdue : la bande jaune recouvre toutes les autres, dans une région où l'on aurait dû avoir des points frontières. (b) L'enveloppe visuelle d'un tore avec quatre points de vue. A noter les ramifications de la bande rose.

$v \neq i$ et j, tout comme une arête d'intersection de cône.

3. Un point triple se projette sur trois contours occultants de vues différentes i, j, et k, et à l'intérieur des silhouettes dans toute vue $v \neq i$,j et k.

Nous avons approfondi la structure de l'enveloppe visuelle polyédrique, dans les conditions induites par une discrétisation. Ces connaissances vont permettre de créer des algorithmes pour en construire la représentation. Les algorithmes que nous proposons partent du constat que les segments de vue sont une représentation initiale partielle de l'enveloppe visuelle polyédrique, sous forme d'un sous ensemble des arêtes de ce polyèdre. Les paragraphes suivants précisent comment calculer cette représentation initiale.

3.4 Calcul des segments de vue

Les algorithmes de modélisation que nous présentons dans cette thèse se servent des segments de vue comme étape initiale pour le calcul de l'enveloppe visuelle. Les segments de vue constituent en effet une représentation partielle du maillage polyédrique, contenant déjà beaucoup d'informations sur la surface finale : il s'agit en effet d'un sous-ensemble des arêtes de l'enveloppe visuelle

polyédrique en question.

Nous présentons ici un algorithme pour calculer les segments de vues à partir de N silhouettes polygonales obtenues à partir de caméras calibrées. Ce calcul a déjà été étudié indirectement par Matusik *et al.* [74] qui se sert d'un calcul similaire pour savoir si la ligne de vue d'un pixel intersecte l'enveloppe visuelle, dans le cadre du rendu d'une l'enveloppe visuelle à partir d'un nouveau point de vue ("Image-based Visual Hulls"). Il est aussi utilisé par Cheung *et al.* pour contraindre l'alignement d'enveloppes visuelles au cours du temps à partir de données photométriques [21]. Nous présentons ici un algorithme similaire qui présente l'avantage de s'appuyer sur la formulation ensembliste de l'enveloppe visuelle dérivée au paragraphe 3.2.4, et calculons sa complexité.

3.4.1 Description de l'algorithme

L'algorithme consiste à rechercher, pour la ligne de vue issue d'un point d'un contour dans les images, sa contribution à la surface de l'enveloppe visuelle. Il nécessite donc de rechercher l'intersection de cette ligne de vue avec les cônes visuels de toutes les autres images. Chaque jeu d'intersections avec un des cônes visuels prend la forme d'une liste d'intervalles le long de la ligne de vue en question. Reste ensuite à fusionner ces intervalles issus de toutes les autres images, de sorte à obtenir au final une primitive appartenant à l'intersection de tous les autres cônes et donc à la surface de l'EVP.

Nous résumons l'algorithme ci-dessous (nous notons $\mathcal{L}_j^i$ la ligne de vue associée au point p_j^i) :

Algorithme 1 Calcul des segments de vue

1: **pour chaque** contour O_j^i dans toutes les images : **faire**
2: **pour chaque** image k telle que $k \neq i$: **faire**
3: **pour chaque** point p_j^i dans O_j^i : **faire**
4: **calculer** la ligne épipolaire l de p_j^i dans l'image k,
5: **calculer** les intersections de l avec tous les contours O_l^k de l'image k,
6: **mettre à jour** les intervalles de profondeur le long de $\mathcal{L}_j^i$,
7: **fin pour**
8: **calculer** les points 3D délimitant les intervalles le long de $\mathcal{L}_j^i$.
9: **fin pour**
10: **fin pour**

Calcul des intervalles issus de l'intersection avec un cône de vue

Soit p_j^i un point d'un contour occultant $\mathcal{O}_j^i$. Les intervalles de contribution sur la ligne de vue de p_j^i sont délimités par les intersections de cette ligne de vue avec

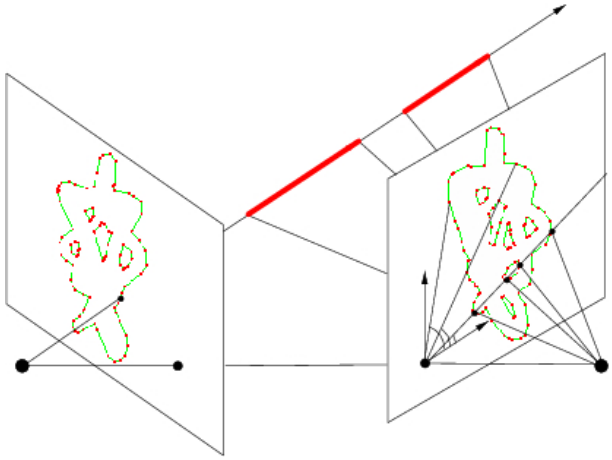


FIG. 3.13 – Intervalles de contribution à l'enveloppe visuelle (en rouge) le long de la ligne de vue.

les surfaces des cônes de vue concernés. On peut déterminer ces intervalles en effectuant des calculs d'intersection de droite et de cône dans l'espace. Cependant il est plus judicieux d'effectuer les opérations équivalentes en 2D directement dans les images comme suggéré par Matusik *et al.*[74]. En effet, les bornes de l'intervalle sur la ligne de vue de p_j^i peuvent être déterminées à l'aide de la géométrie épipolaire (voir figure 3.13).

Mise à jour des intervalles de profondeur

Une fois les intervalles de profondeurs obtenus à partir pour chaque cône visuel, il reste à combiner les listes de profondeurs obtenues avec des contours d'images différentes.

Nous procédons de la manière suivante : l'expression (3.2) est utilisée pour fusionner sur l'ensemble des contours et des images les intervalles contribuant au complémentaire de l'enveloppe visuelle. L'utilisation de cette formule, telle qu'elle est écrite, nécessite de regrouper les contours d'une silhouette par composante (un contour extérieur et les contours intérieurs qui lui sont associés), ce qui peut s'avérer fastidieux. Mais il est possible de travailler avec tous les contours de manière symétrique, sans regroupement. En effet, la contribution des contours intérieurs appartient déjà au complément de l'enveloppe visuelle. C'est pourquoi on peut se contenter d'effectuer l'opération d'intersection sur les contributions des contours extérieurs seulement. Voici l'expression correspondante obtenue à partir de (3.2) :

$$\mathcal{VH}^c(\mathcal{I},\mathcal{K}) = \bigcup_{i\in\mathcal{I}} \left[\left(\bigcap_{j\in \text{Exterieurs}^i} \mathcal{D} \setminus \mathcal{V}^i_j \right) \bigcup \left(\bigcup_{j\in \text{Interieurs}^i} \mathcal{D} \setminus \mathcal{V}^i_j \right) \right] \tag{3.5}$$

où Exterieurs^i et Interieurs^i sont les ensembles de contours extérieurs et intérieurs de l'image i. L'expression obtenue est équivalente à (3.2) mais simplifie la fonction de mise à jour des intervalles. Notons que d'après les définitions données de $\mathcal{D}$ et $\mathcal{V}^i_j$, la contribution le long de la ligne de vue du complémentaire de chaque cône de vue doit être limitée à l'intervalle visible depuis l'image qui lui correspond. Ceci permet de rendre compte d'objets non visibles par toutes les caméras, comme précédemment expliqué.

3.4.2 Complexité

Soient n le nombre d'images, m le nombre d'objets dans la scène, q le nombre maximal de points par contour dans les images. Pour simplifier l'étude, nous nous intéressons à un cas particulier : celui où les m objets observés dans la scène sont convexes. Ceci est suffisant pour rendre compte du comportement de l'algorithme, tout en simplifiant l'analyse des ambiguïtés visuelles, qui ne peuvent alors être dues qu'à des phénomènes inter-objets. Cette étude se généralise à une scène quelconque en décomposant celle-ci en un nombre minimal de régions convexes. En effet les ambiguïtés visuelles inhérentes aux silhouettes sont dues aux non-convexités de l'ensemble observé, qui peuvent aboutir à un comportement quadratique de l'algorithme.

Nombre de points calculés L'algorithme proposé calcule au maximum $O(nm^2q)$ points 3D pour le cas de m objets convexes. L'aspect quadratique du terme en m de cette complexité est dû à la possible apparition d'objets fantômes : en effet dans les pires cas d'ambiguïté visuelle m^2 objets peuvent être reconstruits dans la scène alors que seuls m objets sont physiquement présents (voir figure 3.14). En revanche le nombre de points calculés est bien linéaire en n et q : chaque objet convexe reconstruit dans la scène comporte en effet n bandes comportant chacune $O(q)$ primitives, et le nombre d'objets fantômes ne dépend ni du nombre de points de vue[1], ni du nombre de sommets des contours occultants[2].

Nombre d'opérations Le calcul des segments de vue consiste en un ensemble de sous-calculs effectués pour chaque ligne de vue. Le nombre de lignes de vue

[1] le maximum d'objets fantômes peut en effet déjà être atteint pour deux vues, mais ne peut que diminuer avec l'adjonction de nouvelles vues.

[2] grâce à l'hypothèse de convexité des objets. Dans le cas contraire des objets fantômes pourraient apparaître du fait des ambiguïtés visuelles se produisant au sein d'un même objet.

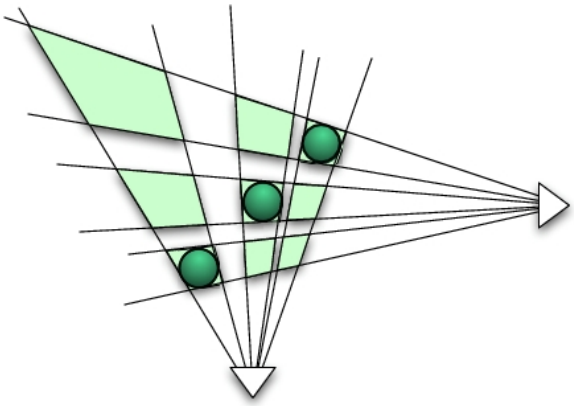


FIG. 3.14 – Illustration, avec deux vues, du comportement quadratique avec le nombre d'objets réel présents dans la scène, inhérent à l'ambiguïté visuelle dues à l'observation de silhouettes. Les trois sphères réelles donnent lieu à neuf objets reconstruits dont six fantômes.

est $O(nmq)$: il y a en effet une ligne de vue par sommet de contour, donc $O(q)$ lignes de vue par contour, avec $O(m)$ contours dans chacune des n images. L'ensemble des primitives de la scène est donc calculé avec $O(nmqr)$ opérations, où r est le majorant du nombre d'opérations nécessaires pour calculer toutes les intersections possibles entre une ligne de vue et les cônes de vue issus des contours des autres images. Une implémentation naïve engendrerait un algorithme d'intersection de complexité $r = O(nmq)$ (c'est à dire pour chaque ligne de vue, chercher une intersection avec toutes les arêtes de tous les contours des autres images). Le nombre d'opérations de l'ensemble serait alors de $O(n^2m^2q^2)$.

Il apparaît cependant que la complexité r peut être réduite à $O(n \log mq + a)$, a étant le nombre d'intersections trouvées sur une ligne de vue. A cet effet il est possible de rectifier l'image k de sorte que les lignes épipolaires deviennent horizontales, auquel cas la recherche d'intersections entre une ligne épipolaire l et un ensemble de contours occultants est simplifiée, en indexant la recherche avec l'ordonnée des épipolaires. Ainsi, déterminer quels segments du contours sont intersectés revient à rechercher ceux dont l'ordonnée des points qui les définissent bornent l'ordonnée de l'épipolaire. On peut utiliser de manière équivalente les angles des droites reliant l'epipole aux points du contour pour indexer la recherche, ce qui évite avantageusement la rectification épipolaire (voir figure 3.13). Chacune de ces solutions permet d'arriver à $r = O(n \log mq + a)$, moyennant l'ajout d'un nombre d'opérations $O(n^2mq \log mq)$ pour construire la structure d'indexation dans les deux cas. Notons que [74, 73] utilisent la pente des lignes épipolaires dans un but identique, mais la partition du plan qui en résulte peut s'avérer moins pratique dans la mesure ou une classification erronée peut avoir lieu au voisinage

de la droite verticale incidente à l'epipole, endroit où la fonction de calcul de la pente est non monotone et discontinue. L'utilisation d'angles nécessite aussi de gérer un cas de discontinuité à cause de leur congruence modulo 2π.

Pour chacune des $O(nmq)$ lignes de vues, a intersections sont collectées dans toutes les autres images. Le nombre de ces intersection est $a = O(nm)$ car pour chacune des n images, $O(m)$ intersections sont possibles étant donné que $O(m)$ contours convexes apparaissent dans chaque image. L'expression de mise à jour (3.5) est appliquée pour calculer les segments de vue à partir de ces $O(nm)$ intersections. La fusion des intervalles s'apparente à une fusion de listes triées, nécessitant $O(nm)$ opérations. Cette fusion engendre la création de $O(m)$ segments de vue pour chacune des lignes de vues, qui aboutit au nombre total de primitive en $O(nm^2q)$.

Le nombre total d'opérations de l'algorithme est la somme de deux termes, le terme d'initialisation avec $O(n^2mq \log mq)$, et le terme de calcul $O(nmq(n \log mq + nm))$. La complexité dominante dans les deux termes est $O(n^2mq \log mq)$. Le calcul des segments de vue proposé est donc quadratique avec le nombre d'images et quasi-linéaire en mq. Heureusement dans notre contexte il est rare d'utiliser plus de quelques dizaines ou centaines d'images. Nous constatons aussi une quasi-linéarité de l'algorithme en fonction du nombre de sommets des contours. Contrairement au nombre d'images, il est possible d'exercer un certain contrôle sur q, via les possibilités de simplification évoquées au chapitre 2.

3.5 Algorithme hybride

Nous avons précédemment montré comment calculer des points sur la surface d'une enveloppe visuelle polyédrique, grâce au calcul des segments de vue. Nous nous intéressons maintenant à une première méthode d'estimation de la forme de l'enveloppe visuelle. Les approches volumétriques classiques reposaient sur une partition régulière de l'espace en cellules élémentaires : les voxels. Nous proposons ici une partition s'appuyant sur les sommets des segments de vue calculés ci-dessus, et qui se compose de cellules non régulières : les tétraèdres de Delaunay. L'avantage réside dans le fait que l'on conserve une forte précision grâce à un pouvoir accru de représentation des surfaces, pour une complexité spatio-temporelle raisonnable : l'algorithme propose un compromis bénéfique entre approches volumétriques et approches surfaciques.

Ces travaux ont été réalisés en collaboration avec Edmond Boyer, et publiés dans [16] et [38].

3.5.1 Triangulation des points

Les triangulations de Delaunay ont déjà été vastement utilisées pour reconstruire des surfaces 3D à partir de points 3D quelconques. Ce problème a été exhaustivement étudié dans la précédente décennie, et la plupart des méthodes proposées considèrent que la surface recherchée est contenue dans la triangulation de Delaunay des points. La triangulation de Delaunay comporte deux avantages : d'une part elle assure une partition régulière de l'espace avec des cellules possédant des propriétés démontrées ; d'autre part, il en existe des implémentations rapides et robustes.

Le problème que nous considérons est semblable mais comporte en entrée, en plus des points 3D, l'information 2D des images. Ainsi notre approche recherche également un sous-ensemble de la triangulation de Delaunay, et se base donc sur un critère image pour sculpter les cellules tétraèdriques incohérentes. D'un point de vue complexité, il est bien connu le cas le pire de la tétraèdrisation de Delaunay est en $O(N^2)$, avec N le nombre de points [32]. En ce qui nous concerne, comme nous calculons $O(nm^2q)$ points, avec n le nombre d'images, m le nombre moyen de contours, et q le nombre moyen de points par contour, la borne supérieure de la complexité de notre algorithme serait alors en $O(n^2m^2q^2)$, ce qui est supérieur au temps passé à obtenir les points de la surface de l'enveloppe visuelle. Cependant des travaux récents [9] tendent à montrer que la complexité de la triangulation de Delaunay est linéaire pour des points disposés sur un polyèdre. Ceci est par ailleurs confirmé par nos résultats expérimentaux qui montrent que le calcul des points sur la surface n'est pas dominé par celui de la triangulation de Delaunay en complexité.

3.5.2 Extraction de la surface

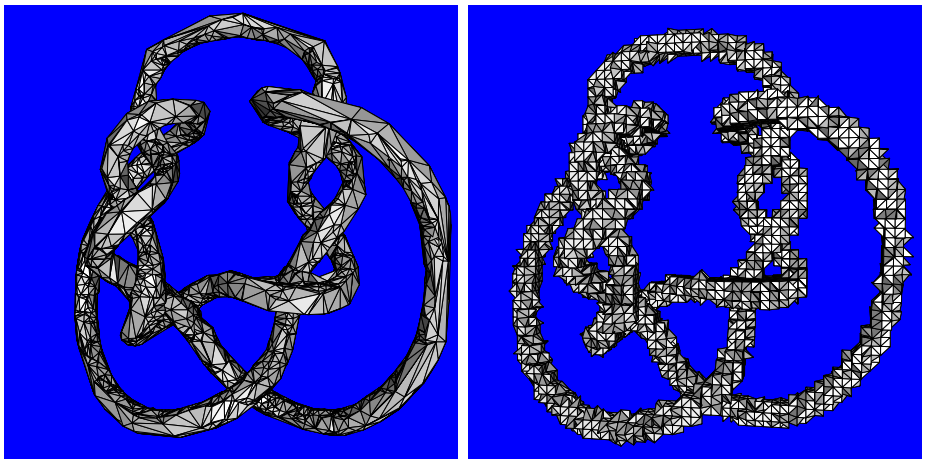
La triangulation de Delaunay réalisée sur les sommets de segments de vue engendre un ensemble de tétraèdres dont l'union forme l'enveloppe convexe des points fournis. Nous devons maintenant identifier et éliminer ceux qui contribuent au complément de l'enveloppe visuelle. Une méthode ad hoc consiste à vérifier si le centre des tétraèdres se projette à l'intérieur de toutes les silhouettes. Une fois les tétraèdres sculptés, il suffit alors d'extraire la surface de cette représentation, c'est à dire l'ensemble des facettes de tétraèdres se trouvant à la limite entre les volumes intérieurs et extérieurs identifiés. La surface extraite est donc un polyèdre non dégénéré, grâce aux bonnes propriétés de la tétraédrisation de Delaunay, et sous réserve d'éliminer les tétraèdres de manière à conserver une variété (l'élimination imprudente de certains tétraèdres qui partagent un seul point ou une seule arête avec un autre tétraèdre éliminé peut engendrer une surface dégénérée). Cette approche est rapide dans le cas où l'on dispose d'images binaires indiquant pour chaque pixel son appartenance ou non à l'arrière plan, ce qui est souvent le cas dans

le cadre d'applications basées sur les silhouettes. Aussi, notre expérience montre que cette méthode donne des résultats satisfaisants. Notons que des critères d'acceptation plus complexes mettant en jeu la surface ou le volume du tétraèdre sont également envisageables mais n'ont pas été explorés au cours de cette thèse. Il serait par exemple facile d'adapter la stratégie d'échantillonnage proposée par Kanade *et al.* [22], faisant intervenir plusieurs points appartenant au volume d'un tétraèdre.

3.5.3 Résultats expérimentaux

Nous avons appliqué notre méthode à divers jeux de données. Une première expérience consiste à comparer notre approche aux approches à base de voxels. La figure 3.15 montre des résultats obtenus à partir de silhouettes identiques (40 images). Les bornes de la grille des voxels ont été choisies proches de la surface de l'objet, ce qui est rarement le cas pour des applications réelles. Notons que les résultats sont géométriquement meilleurs pour une complexité sensiblement inférieure. Notre modèle comporte en effet 3772 points, alors qu'il faut vérifier $60^3 = 216000$ voxels avec chaque image avec l'autre approche, sans parler d'une éventuelle étape d'extraction de surface. Par ailleurs, le nombre d'images a une influence linéaire sur la borne supérieure de la complexité de l'approche volumétrique, alors que cette influence est quadratique pour notre méthode. En effet le partitionnement de l'espace est indépendant du nombre d'images pour les approches volumétriques, ce qui n'est pas le cas pour notre approche. Notons cependant que l'addition d'images supplémentaires ne permet pas forcément d'améliorer l'estimation de la surface, comme le montrent nos prochaines expériences.

Une seconde série de résultats a été obtenue à partir d'images d'un tore de synthèse et de points de vue aléatoirement distribués sur une sphère entourant ce tore. La figure 3.16 montre différentes enveloppes visuelles du tore obtenues en faisant varier le nombre de points échantillonnés sur les contours 2D des images ainsi que le nombre de caméras. Notons que le temps d'exécution de l'algorithme est en $O(n^2m^2q)$ avec n, m, q le nombre d'images, d'objets dans la scène, et de points par contour respectivement. L'addition de points échantillonnés sur les contours 2D a donc un impact moins sensible sur le temps d'exécution. Il est surprenant de constater qu'à partir d'un certain nombre d'images ajoutées, la précision du modèle d'enveloppe visuelle calculé décroît. Ce phénomène est visible dans la colonne de gauche, au passage de 16 à 32 images. Il s'agit d'un problème de sous échantillonnage : si l'on augmente le nombre de points de vues, il faut aussi significativement augmenter le nombre de points de la triangulation pour permettre à l'algorithme de capturer la géométrie de la reconstruction. La parade à ce défaut est donc d'imposer l'augmentation de points échantillonnés sur le contour 2D lorsque le nombre d'images augmente, pour fabriquer davantage de segments de vue et réduire cette erreur. Il est intéressant de constater que cela



(a) (b)

FIG. 3.15 – La surface de l'enveloppe visuelle pour les nœuds : (a) résultat de notre algorithme (3772 points reconstruits) (b) une reconstruction voxelique sur une grille de 60x60x60.

suggère l'existence d'un rapport optimal entre le nombre d'images et le nombre de points pris sur les contours.

En ce qui concerne le temps d'exécution, facteur crucial pour les applications temps réel, notre approche calcule le premier modèle de tore en 0.008s et le dernier modèle en 1.3s sur un seul PC à 1.8GHz. Cependant les résultats sur un seul PC avec plus de 4 caméras temps réel ne sont pas vraiment significatifs en pratique. Sur le modèle des nœuds, et avec 4 caméras aléatoirement placées, le temps d'exécution moyen est de 0.45s pour un modèle final comportant 2200 points. Comme il a précédemment été expliqué, l'algorithme passe la majorité de son temps dans la phase de calcul des points sur la surface de l'enveloppe visuelle, et pas sur la triangulation de Delaunay, qui traite les 2200 points du modèle en 0.3s. En outre la complexité linéaire de cette opération dans le contexte de sommets pris sur un polyèdre est vérifiée dans nos résultats.

Un autre exemple, illustré en figure 3.17 illustre le potentiel de cette méthode dans le cadre de l'estimation de surfaces courbes. La géométrie de l'objet, une cruche, est fidèlement restituée dès qu'une dizaine de vues est disponible.

3.6 Discussion

Nous avons proposé un premier algorithme pour reconstruire la forme d'objets d'intérêt à partir de silhouettes. Il présente une contribution significative, en tant que nouvel algorithme d'estimation de surfaces à partir de silhouettes, présentant un compromis entre les approches volumétriques et des approches sur-

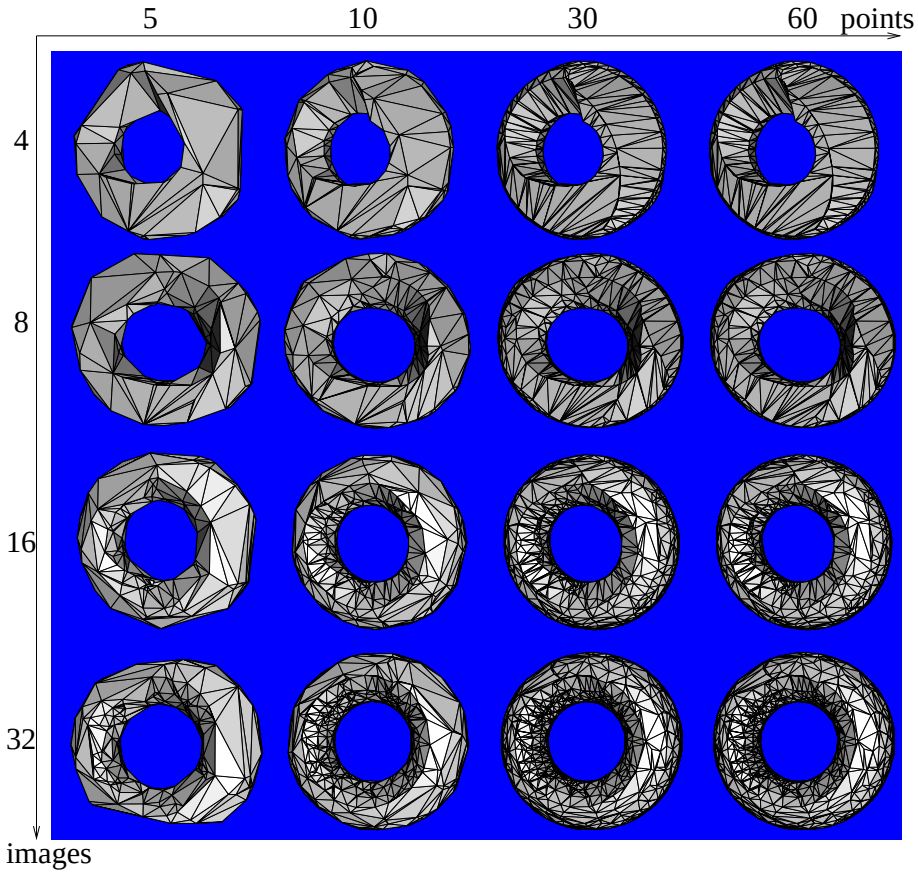


FIG. 3.16 – Enveloppes visuelles d'un tore : en abscisse le nombre de points pris sur le contour et en ordonnée le nombre d'images. Les points sont régulièrement échantillonnés sur les contours et les images aléatoirement choisies sur une sphère entourant le tore.

faciques. Nous avons montré que notre approche est équivalente aux approches volumétriques en terme d'efficacité. Nous avons aussi montré que notre approche donne des résultats significativement meilleurs en terme de précision, tout en ayant une complexité plus réduite dans le temps et l'espace.

L'algorithme commence par calculer des points sur la surface de l'enveloppe visuelle, puis extrait la surface de l'enveloppe visuelle en partant d'une triangulation de Delaunay. Il présente ainsi l'avantage de déléguer tous les problèmes liés à la création de surface (évoqués au paragraphes 3.1.3 et 3.1.4) à des librairies spécialisées, bénéficiant ainsi des résultats sur les problèmes numériques et géométriques liés à la triangulation de Delaunay, qui ont été très étudiés. Plusieurs libraires de ce type existent [2, 6, 4]. L'algorithme permet ainsi de répondre efficacement au problème de la modélisation de la surface de l'enveloppe visuelle sous forme de polyèdre fermé et sans auto-intersection.

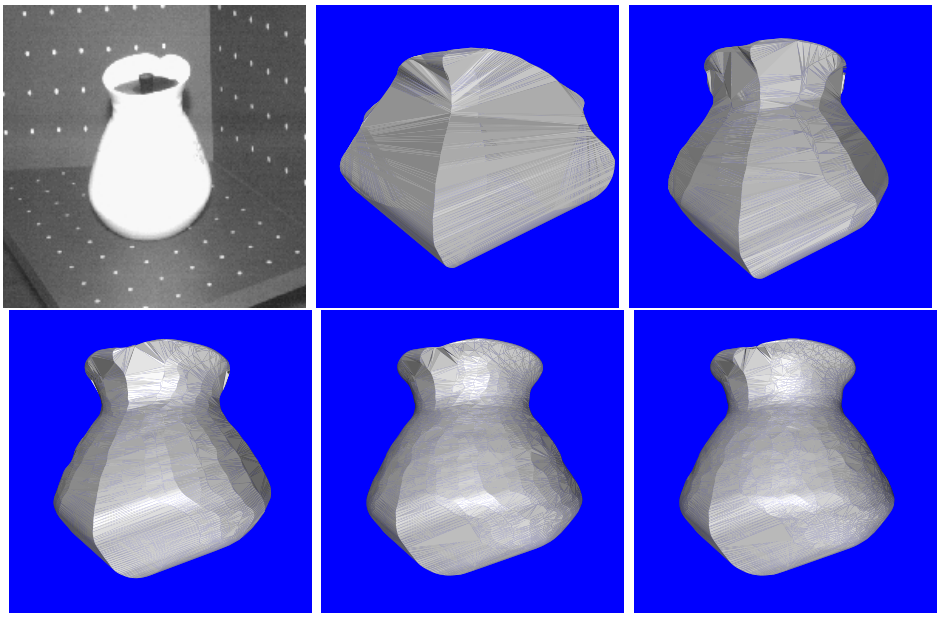


FIG. 3.17 – L'enveloppe visuelle d'une cruche avec 2, 5, 10, 20 et 30 images. Certains cônes grossiers sont encore visibles même avec 30 images du fait du manque de vues dans une large région derrière l'objet.

Il est important de garder en tête que cet algorithme engendre, au même titre qu'un algorithme à base de voxels, une approximation de l'enveloppe visuelle. Il s'agit cependant, comme l'illustrent les résultats pratiques, d'une très bonne approximation de celle-ci. Cette représentation est tout de même incomplète : l'information que nous fournissons à l'algorithme sur l'enveloppe visuelle, les segments de vues, est insuffisante pour une détermination complète du polyèdre de l'enveloppe visuelle. Les points triples ne sont en effet pas représentés parmi les points des segments de vue.

Nous n'utilisons pas non plus le fait que chaque segment de vue est lui-même sur la surface de l'enveloppe visuelle. Nous pourrions prendre en compte cette information à l'aide d'une tétraèdrisation de Delaunay contrainte [23], mais de telles méthodes sont très lourdes en temps de calcul et risquent d'être incompatibles avec le temps réel. Dans tous les cas, l'ajout de contraintes ne permettrait pas de garantir l'extraction du polyèdre complet de l'enveloppe visuelle avec certitude : il faudrait pour cela contraindre chaque point et chaque arête de la décomposition intervenant dans le polyèdre final, ce qui suppose déjà la connaissance totale de ce polyèdre. C'est pourquoi nous nous penchons, jusqu'à la fin de ce chapitre, sur l'étude d'un algorithme permettant d'extraire directement une enveloppe visuelle polyédrique.

3.7 Enveloppe visuelle exacte

Au vu des résultats existants, il apparaît que des algorithmes très divers pour reconstruire l'enveloppe visuelle d'un objet à partir de silhouettes ont été proposés. Cependant peu de méthodes existantes permettent de calculer une surface de l'enveloppe visuelle à la fois précise et dont la surface est une variété, de manière rapide et robuste. Tout en poursuivant l'analyse des conditions à réunir pour créer un tel algorithme, nous proposons ici une nouvelle solution, une approche gloutonne pour calculer une représentation polyédrique équivalente à l'enveloppe visuelle, correspondant à un jeu de silhouettes perturbées donné. L'algorithme présenté possède les avantages recherchés : il s'écrit de manière relativement simple et ne porte à tout moment que sur une arête du polyèdre. Tout le problème d'intersection de cônes visuels est ramené à un ensemble de calculs à inconnues unidimensionnelles pouvant tous être effectués dans les images : l'ordonnancement d'intersections le long de la direction de l'arête recherchée. Nul besoin donc de se ramener à un algorithme générique d'intersections de primitives 2D ou 3D : le problème peut être résolu directement. De ce fait l'analyse des propriétés de robustesse et de convergence de l'algorithme peut être effectuée dans de très bonnes conditions. Nous calculerons également la complexité en temps de calcul de cet algorithme et le comparerons avec les méthodes de référence.

3.7.1 Motivation

L'idée de cet algorithme provient de plusieurs constats :

- Les segments de vue, qui sont créés à partir de points des contours de silhouettes dans les images, constituent de ce fait des arêtes du polyèdre recherché.
- Si l'on calcule les segments de vue à partir des sommets de contours polygonaux obtenus par vectorisation de silhouettes, alors toute primitive du polyèdre de l'enveloppe visuelle se projette sur des arêtes de ces contours.
- Le calcul des segments de vue nous fournit des sommets et des arêtes du polyèdre final, mais il peut fournir plus : on sait quelle arête de quelle image a contribué à engendrer chacun de ces sommets. Chaque point des segments de vue calculé possède donc une information partielle d'incidence, par rapport aux autres primitives de la surface. Comme nous le montrerons au cours de ce chapitre, cette information est suffisante pour retrouver, de proche en proche, toute l'information d'incidence et toute primitive du maillage.

Nous décrirons l'algorithme lui-même en section 3.8, en commençant par le cas, plus simple (paragraphe 3.8.1), ne faisant intervenir que deux images, puis nous

expliquerons l'algorithme proposé dans le cas le plus général (paragraphe 3.8.2). Auparavant, nous précisons comment construire un maillage cohérent à partir de l'orientation des primitives dans les images. Nous analyserons l'algorithme et calculerons sa complexité, puis nous montrerons des résultats de l'algorithme et une comparaison avec les méthodes de référence. Ces travaux ont été réalisés en collaboration avec Edmond Boyer [39].

3.7.2 Incidences et orientations des primitives du maillage

Pour engendrer une description correcte du polyèdre associé à un jeu de silhouettes, il faut conserver au cours du processus de reconstruction une information d'incidence et d'orientation locale cohérente. Nous nous intéressons donc aux relations d'incidence entre les différentes primitives du maillage, en l'occurrence le sommet, l'arête, et la facette.

Nous présentons d'abord la structure de données utilisée pour conserver ces informations d'incidence. Nous nous intéressons ensuite au lien entre les primitives des silhouettes et celles du maillage, qui en héritent l'orientation.

Structure de données utilisée

Parmi les structures de données existantes, celle des *Winged Edges* définie par [12] est la plus utilisée pour maintenir l'état des adjacences sur la surface dans les divers algorithmes de manipulation des maillages. Cette structure met l'arête au centre de la structure de données, et stocke l'information d'incidence à huit primitives voisines (voir figure 3.18) : les deux facettes de chaque côté de l'arête, les deux sommets incidents, ainsi que les quatre arêtes voisines pour l'orientation.

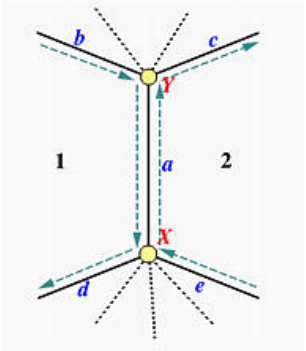


FIG. 3.18 – Structure des *Winged Edges*

En partant de l'hypothèse que notre polyèdre est constitué uniquement de points trivalents (argumentée en 3.3.3, et illustrée figure 3.19(b)), nous optons pour une simplification de cette structure (figure 3.19(a)), sous la forme d'un graphe orienté dont les sommets sont ceux du maillage. Chaque sommet stocke l'incidence à ses trois arêtes et trois facettes voisines. La représentation des arêtes est donc la seule redondante dans ce cas, puisqu'elle figure implicitement dans les arcs orientés du graphe, pouvant provenir de l'un ou l'autre des sommets incidents à l'arête. C'est pourquoi l'on manipule toujours une arête "orientée" du maillage, selon le sens de parcours donné par la tâche en cours.

Grâce à la structure de données proposée, les relations d'incidence et d'orientation entre les primitives du maillage restent simples à gérer. Pour une arête

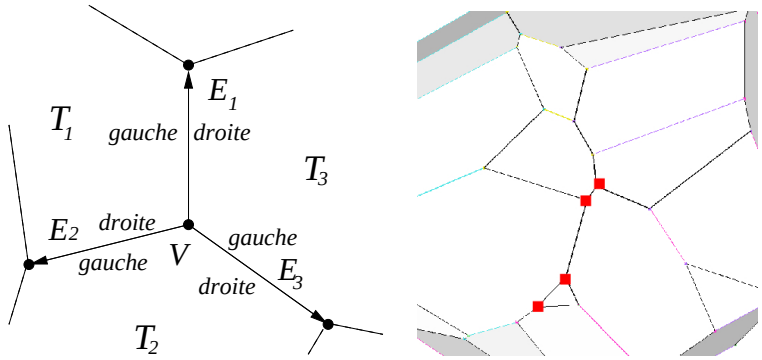


FIG. 3.19 – (a) La relation entre l'orientation du sommet V (l'ordre des paires $(E_i, \mathcal{T}_i)$) et l'orientation (gauche et droite) de chaque arc partant de V sur la surface du polyèdre. (b) Une vue rapprochée de l'enveloppe visuelle d'une sphère. Les arêtes en gras sont des courbes d'intersection discrètes de cônes, les points marqués par des carrés sont des points triples. La structure du maillage est mise en évidence, en particulier la trivalence des sommets de celui-ci.

orientée du maillage, l'information d'orientation se réduit à la connaissance de ce qui est à gauche et à droite de l'arête sur la surface ; pour les sommets, elle se résume à savoir comment sont ordonnées les trois arêtes qui lui sont incidentes.

Lien entre les primitives image et celles du maillage

Considérons tout d'abord une arête orientée E du polyèdre de l'enveloppe visuelle. Une telle arête délimite deux facettes polygonales de la surface de l'enveloppe visuelle, une localement à sa gauche et l'autre localement à sa droite. Comme mentionné plus tôt, ces deux facettes sont chacune incluses dans une facette de cône engendrée par une arête d'un contour. Notons ces facettes de cône respectivement $\mathcal{T}_{gauche}$ et $\mathcal{T}_{droite}$. Nous prenons donc comme convention de conserver dans la structure de données l'adjacence entre les sommets et arêtes du maillage, et les facettes de cône $\mathcal{T}$ mentionnées. Les facettes de cône donnent en effet la géométrie des plans adjacents à ces primitives, et permettent également de conserver l'association avec la primitive image desquels ils sont issus.

Notons que pour les segments de vue, cas particulier d'arêtes du maillage de l'enveloppe visuelle, la paire $(\mathcal{T}_{gauche}, \mathcal{T}_{droite})$ peut être identifiée grâce à l'orientabilité des bandes héritée des contours image, comme le suggère la figure 3.20. Ceci permet d'initialiser correctement la structure à partir des segments de vues initiaux.

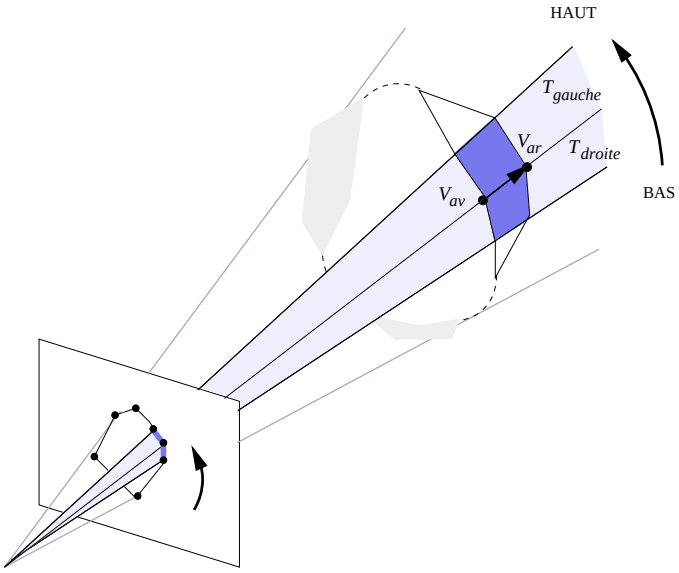


FIG. 3.20 – Relation entre les incidences de l'arête orientée $[V_{av}, V_{ar}]$ correspondant à un segment de vue, et les faces de cône adjacentes, et lien avec l'orientation de la bande et de l'image.

3.8 Description de l'algorithme

En nous appuyant sur les propriétés des enveloppes visuelles polyédriques évoquées, et les relations entre les primitives qui les constituent, nous pouvons maintenant décrire l'algorithme lui-même. Nous commençons par un cas simple, celui de deux images, où l'absence de points triples facilite la recherche des arêtes résultant d'intersections de cônes et permet de fournir une solution simple. Nous traiterons ensuite le cas général, où la géométrie des intersections de cônes est plus complexe.

3.8.1 Cas de deux silhouettes

Penchons nous sur le cas de l'enveloppe visuelle polyédrique obtenue avec deux images. C'est un cas simple intéressant dans la mesure où le calcul des segments de vues fournit la quasi-totalité de l'information pour calculer le polyèdre : en effet la détermination des arêtes d'intersection de cônes ne nécessite aucun calcul géométrique supplémentaire. Dans ce cas particulier, seules quelques déductions algorithmiques sont nécessaires pour compléter le maillage, comme nous allons le montrer. Le cas à deux images fournit par ailleurs certaines clés pour la compréhension du cas général à N images.

Examinons tout d'abord quelles primitives image contribuent à la formation d'une arête d'intersection de cône E. Comme nous l'avons vu précédemment, une

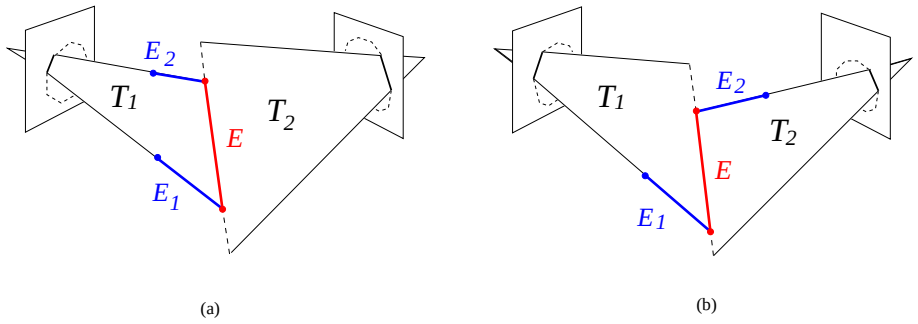


FIG. 3.21 – Formation d'une arête d'intersection de cône dans le cas de deux images, faisant apparaître les segments de vue impliqués E_1 et E_2.

telle arête résulte de l'intersection de deux facettes de cône, issues de l'une et de l'autre image. Ceci est illustré en figure 3.21. Seuls deux cas de figure sont possibles, en supposant l'absence de cas dégénérés et la trivalence de tous sommets :

- (a) les sommets de l'arête E sont sur deux lignes de vues provenant d'une même image.
- (b) les sommets de l'arête E sont sur deux lignes de vues provenant de l'une et l'autre image.

Dans tous les cas, les sommets de l'arête sont sur deux lignes de vue. Ce qui signifie notamment que tous ces sommets sont déjà calculés au cours de l'étape d'obtention des segments de vue. Pour finir le calcul du polyèdre, il suffit alors d'identifier toutes les arêtes E correspondant à ces cas de figure.

Pour ce faire, au cours du calcul des segments de vue, il faut stocker pour chaque sommet quelle facette de cône a engendré le sommet, par intersection avec la ligne de vue de celui-ci. Dans le cas de la figure 3.21(b) par exemple, le segment de vue E_1 comporte un sommet engendré par intersection de la ligne de vue de E_1 et de la face triangulaire infinie $\mathcal{T}_2$. Réciproquement le segment de vue E_2 comporte un sommet engendré par intersection avec $\mathcal{T}_1$. La détection de cette réciprocité est suffisante pour identifier les sommets de l'arête E. Le cas de la figure 3.21(a) est lui aussi facilement détectable, car les deux segments de vue E_1 et E_2 ont tous deux été engendrés par intersection de leur ligne de vue respective avec la facette de cône $\mathcal{T}_2$.

Dans tous les cas la recherche d'une arête d'intersection de cône donnée ne fait donc intervenir que les quatre lignes de vue incidentes à $\mathcal{T}_1$ et $\mathcal{T}_2$, et peut se faire par un parcours des lignes de vues concernées. La figure 3.22 montre un exemple de résultat obtenu pour la reconstruction d'un tore via cette recherche d'arêtes d'intersection de cônes. Dans cet exemple, la position des caméras est telle que quatre branches apparaissent à cause de l'ambiguïté visuelle. Deux branches contiennent effectivement la véritable surface du tore. Les deux autres branches ne renferment pas de matière réelle mais se projettent néanmoins à l'intérieur de

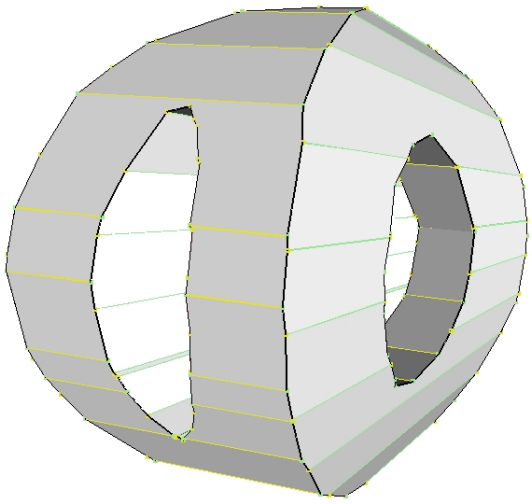


FIG. 3.22 – Enveloppe visuelle polyédrique d'un tore obtenue avec deux silhouettes avec le schéma de recherche évoqué. Le segments de vue sont en couleurs et les arêtes d'intersection de cônes en noir.

toutes les silhouettes, ce sont des *régions fantômes*. Ceci est conforme au résultat attendu du calcul de l'enveloppe visuelle pour un faible nombre d'images, qui contiennent peu d'informations.

3.8.2 Cas général à N silhouettes

Nous présentons dans cette partie l'algorithme permettant de calculer les arêtes d'intersection de cônes dans le cas général. Pour référence, un résumé de l'algorithme est fourni au paragraphe 3.8.6. Le cas à deux images ne se généralise pas simplement. Dès trois images, une catégorie supplémentaire de points apparaît, les points triples, impliquant par définition trois cônes de vue. C'est à dire qu'il est désormais faux d'affirmer que tous les sommets de l'enveloppe visuelle sont calculés par l'étape des segments de vue, ou encore de supposer que les arêtes d'intersection de cônes prennent toutes les formes triviales de la figure 3.21. Désormais, de telles arêtes peuvent donc joindre des sommets de segments de vue à des points triples, ou encore des points triples entre eux. Elles forment donc un ensemble de graphes connexes, chacun étant composé d'un nombre arbitraire de points triples, reliés à des sommets de segments de vue, comme illustré par la figure 3.19(b).

Néanmoins il est possible de retrouver toutes les composantes inconnues de ces graphes (arêtes d'intersection de cônes et points triples) en partant des segments de vue, avec des calculs d'intersection à une seule inconnue effectués de proche en proche. L'algorithme que nous proposons pour ce faire se décompose en deux tâches. La première est d'identifier, à partir d'un état partiel du maillage, les som-

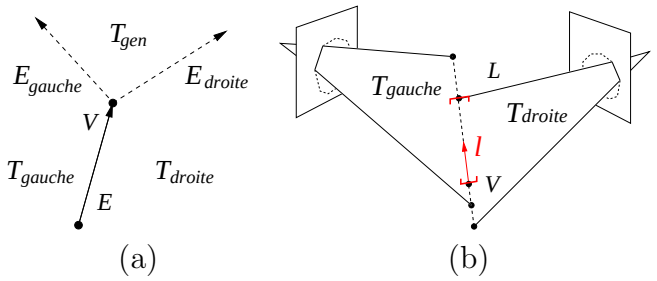


FIG. 3.23 – (a) La recherche d'arêtes manquantes à partir de la situation type rencontrée pendant la construction du maillage d'une enveloppe visuelle polyédrique. (b) Une fois la direction d'une arête manquante connue, la connaissance locale des incidences $\mathcal{T}_{gauche}$ et $\mathcal{T}_{droite}$ de l'arête permet de déduire des bornes de recherche naturelles pour le sommet incident à cette nouvelle arête.

mets incomplets où il manque des arêtes, ainsi et la direction de ces arêtes manquantes, qui est calculable a priori (paragraphe 3.8.3). La deuxième consiste, pour chaque arête manquante, à retrouver le sommet qui se trouve à l'autre extrémité de l'arête, qu'il soit point triple ou sommet d'un segment de vue (paragraphe 3.8.4).

3.8.3 Identification des arêtes manquantes

Les segments de vue rendent compte de toutes les contributions d'une image à l'enveloppe visuelle. Chaque sommet d'un quelconque segment de vue est cependant initialement incomplet, et du fait de la trivalence de ses sommets, nous savons qu'il lui manque exactement deux arêtes. Il suffit d'étudier comment ajouter de nouveaux sommets de proche en proche à cette représentation initiale.

Toute nouvelle arête ajoutée au maillage nous ramène alors au même problème que le problème initial : celui d'un sommet "pendant" connecté à une arête existante, et deux arêtes manquantes en ce sommet. Étudions donc cette configuration en détail.

La situation décrite est schématisée en figure 3.23(a). Un sommet V sur la surface du polyèdre est connecté à une arête E, mais il lui manque encore potentiellement deux arêtes E_{gauche} et E_{droite}. Nous considérons l'arête E orientée en direction de V, c'est à dire dans le sens de parcours du maillage par l'algorithme. Nous supposons également connues les facettes de cône $\mathcal{T}_{gauche}$ et $\mathcal{T}_{droite}$ qui engendrent les facettes du polyèdre de part et d'autre de E. Tout comme dans le cas à deux images, il nous faut conserver toute l'information ayant servie à créer chaque sommet. Ainsi nous supposons connue la troisième facette de cône $\mathcal{T}_{gen}$ qui a permis de d'engendrer V.

Il s'agit maintenant de trouver les arêtes manquantes E_{gauche} et E_{droite}. La re-

cherche ayant pu avoir lieu dans d'autres branches connexes du même graphe de points triples, ces arêtes peuvent déjà avoir été créées, mais une simple vérification dans la structure permet de détecter si tel est le cas. Pour toute arête effectivement manquante, il faut déterminer le sommet du polyèdre sur lequel cette arête débouche, pour pouvoir compléter sa description.

Grâce à l'information d'incidence partielle dont nous disposons, nous connaissons déjà toutes les facettes de cône ayant servi à engendrer V, à savoir $\mathcal{T}_{gauche}$, $\mathcal{T}_{droite}$ et $\mathcal{T}_{gen}$. Nous connaissons donc également grâce à cette information l'incidence des arêtes manquantes. Notamment, l'arête orientée E_{gauche} est incidente aux deux plans de $\mathcal{T}_{gauche}$ et $\mathcal{T}_{gen}$, et E_{droite} est incidente à $\mathcal{T}_{gen}$ et $\mathcal{T}_{droite}$. Nous savons aussi dans ces deux cas que les deux plans en question proviennent toujours d'images différentes : ainsi l'arête elle-même résulte de l'intersection des plans en question. Ceci nous permet de calculer une direction et un sens pour l'une ou l'autre des arêtes manquantes, sous la forme d'un vecteur de l'espace Euclidien, réciproquement $\mathbf{l}_{gauche}$ et $\mathbf{l}_{droite}$. Soient $\mathbf{n}_{gauche}$, $\mathbf{n}_{droite}$ et $\mathbf{n}_{gen}$ les normales aux plans $\mathcal{T}_{gauche}$, $\mathcal{T}_{droite}$ et $\mathcal{T}_{gen}$. Les vecteurs $\mathbf{l}_{gauche}$ et $\mathbf{l}_{droite}$ s'écrivent alors avec des produits vectoriels :

$$\mathbf{l}_{gauche} = k\,\mathbf{n}_{gauche} \times \mathbf{n}_{gen} \qquad \text{avec } k \text{ tel que } |E\ \mathbf{l}_{gauche}\ \mathbf{n}_{gauche}| > 0 \tag{3.6}$$

$$\mathbf{l}_{droite} = k\,\mathbf{n}_{gen} \times \mathbf{n}_{droite} \qquad \text{avec } k \text{ tel que } |\mathbf{l}_{droite}\ E\ \mathbf{n}_{gen}| > 0 \tag{3.7}$$

où le coefficient $k \in \{-1, 1\}$ permet à chaque fois de s'assurer que l'on tourne bien à gauche ou à droite par rapport à E, selon le cas. On définit ainsi, pour chaque arête manquante, une demi-droite de recherche du prochain sommet. Nous décrivons au paragraphe suivant ce qui nous permet de déterminer ce sommet, et le cas échéant, s'il s'agit d'un point triple non encore calculé, comment trouver ses coordonnées.

3.8.4 Sommet incident à une arête incomplète

Nous cherchons maintenant, pour une arête incomplète, le sommet incident de cette arête. Comme nous l'avons fait remarqué, une arête d'intersection de cônes se projette nécessairement sur les contours de deux images, et *à l'intérieur* de toute autre silhouette. Par construction, nous savons déjà sur quels contours se projette l'arête, grâce à l'information d'incidence qui nous a permis d'en calculer la direction. Il nous faut encore trouver le sommet incident S tel que l'arête se projette bien à l'intérieur de toutes les autres silhouettes.

Bornes de recherche naturelles

Notons l'arête incomplète E. Les primitives et incidences en jeu sont schématisées en figure 3.23(b). Y figurent la paire de facettes de cône incidentes

à E, que nous notons $(\mathcal{T}_{gauche}, \mathcal{T}_{droite})$. Nous considérons également la direction de recherche l de l'arête, calculée au paragraphe précédent.

Le fait que l'arête incomplète se trouve sur deux facettes de cône fournit des bornes naturelles à la recherche du sommet incident. Ces bornes sont données par le sommet V à partir duquel on effectue la recherche, et la ligne de vue $\mathcal{L}$ la plus restrictive dans la direction de recherche donnée (voir figure 3.23). Le sommet recherché se trouve nécessairement à l'intérieur de ces bornes. Appelons *arête maximale* l'arête définie par ces bornes de recherche.

Deux cas possibles

Seuls deux cas sont alors à traiter, selon si l'on trouve ou ne trouve pas de cône coupant l'arête maximale :

1. Soit cette arête maximale se projette strictement à l'intérieur de toutes les autres silhouettes. La situation est alors analogue au cas à deux images : l'arête recherchée est strictement la contribution de deux et seulement deux images à l'enveloppe visuelle, et le sommet incident est un sommet de segment de vue. Il peut alors être recherché parmi les segments de vue de la ligne de vue $\mathcal{L}$.
2. Soit il existe un cône visuel qui intersecte l'arête maximale, et l'arête recherchée mène à un point triple. Pour vérifier la contrainte de projection de l'arête à l'intérieur de tous les cônes visuels, il faut choisir parmi ces cônes celui donnant la plus petite arête finale.

Au final, il suffit donc de chercher les intersections possibles avec les cônes visuels autres que ceux de $\mathcal{T}_{gauche}$ et $\mathcal{T}_{droite}$, et de les ordonner le long de la direction l, pour choisir celle donnant la borne la plus restrictive. Si celle-ci est à l'extérieur, alors il n'y a pas de point triple et l'arête maximale est l'arête recherchée de l'enveloppe visuelle. Si celle-ci est à l'intérieure de l'arête maximale, alors nous avons trouvé un point triple dont nous connaissons toutes les incidences et que nous pouvons donc calculer : le point triple se trouve sur le plan des facettes de cône $\mathcal{T}_{gauche}$, $\mathcal{T}_{droite}$ et d'une facette $\mathcal{T}_{gen}$ nouvellement trouvée de la troisième image.

Complétude et terminaison

Le calcul de la nouvelle arête nous ramène au problème du paragraphe 3.8.3. L'algorithme consiste donc naturellement à poursuivre récursivement la recherche dans le graphe de points triples en cours d'exploration. Il est cependant possible que cette exploration mène à un point triple ayant déjà été calculé, dès que le graphe exploré comporte des cycles. Ceci peut se produire lorsque les cotangeances issues de plusieurs images sont très localisées sur la surface. Ces cas sont d'autant

plus fréquents que le nombre de silhouettes utilisées est grand. Il suffit d'indexer les points triples dans une structure de recherche dont la clé est un triplet identifiant les trois faces planes $(\mathcal{T}_{gauche}, \mathcal{T}_{droite}, \mathcal{T}_{gen})$ qui lui sont incidentes. Ceci permet de retrouver un point triple déjà calculé quelle que soit l'arête empruntée pour venir le visiter. La recherche dans cette structure est peu coûteuse car elle peut se restreindre aux seuls points triples du graphe en cours d'exploration et donc rester locale.

Nous avons ainsi défini comment retrouver et créer le graphe complet des intersections de cônes discrets, en partant de la structure des segments de vue initiaux. La complétude des sommets du polyèdre calculés est garantie car l'algorithme calcule à chaque pas une nouvelle primitive qui se trouve par construction sur la surface de l'enveloppe visuelle polyédrique. Aucun sous-ensemble de primitives ne peut être raté car tous les graphes de points triples sont connectés à des sommets de segments de vue, les points de départ de l'algorithme. La terminaison de l'algorithme est garantie car chaque branche d'exploration s'arrête dès qu'elle rencontre un sommet déjà calculé. L'algorithme nous fournit donc en sortie la description complète du maillage d'une enveloppe visuelle polyédrique, toute primitive de ce maillage étant calculée une fois et une seule. Une seule passe supplémentaire est nécessaire pour identifier les facettes polygonales du polyèdre et éventuellement en calculer un maillage triangulé.

3.8.5 Identification des facettes du polyèdre

Le maillage complet du polyèdre associé à un jeu de contours polygonaux étant fourni à l'étape précédente, en tant que graphe d'arêtes orientées, nous pouvons aisément identifier les facettes du polyèdre. Notons que chaque arête 2D dans les images contribue potentiellement à la surface de l'enveloppe visuelle, cette contribution étant un sous ensemble de la facette de cône qui lui est associée. Plus précisément cette contribution est un polygone plan de forme générale, pouvant comporter plusieurs contours intérieurs et extérieurs. Grâce aux propriétés des structures utilisée et décrites en 3.7.2, il nous est possible d'effectuer un parcours de graphe, en tournant à gauche à chaque point rencontré sur la surface, de sorte que le parcours ne concerne que les arêtes connexes d'un même plan, c'est à dire un des contours plans de la facette polygonale recherchée. Chaque arête n'est parcourue que deux fois, une fois dans chaque sens, car elle ne participe qu'à deux tels contours. De ce fait un parcours en $O(S)$ (avec S le nombre de sommets du polyèdre final) nous permet de retrouver l'ensemble complet des contours associé à chaque polygone plan. Notre méthode y trouve un avantage supplémentaire, les polygones en sortie étant décrits dans leur forme la plus concise et générale. Pour les applications qui le nécessitent, un maillage triangulé du polyèdre peut être facilement obtenu en appliquant un algorithme standard [5] de facétisation en $O(v \log v)$ par facette, où v est le nombre de sommets par facettes.

3.8.6 Résumé de l'algorithme

L'algorithme proposé s'articule donc en trois étapes, et fait appel à deux fonctions pour la recherche d'arêtes manquantes et de leur sommet incident. Voici, sous forme algorithmique, le résumé de la méthode proposée :

Algorithme 2 Calcul de l'enveloppe visuelle polyédrique

calculer les segments de vue, en conservant toute info d'incidence
pour chaque sommet V de segment de vue E : **faire**
  cherche_arêtes(V,E)
fin pour
pour chaque arête a de toute silhouette : **faire**
  trouver les contours $\mathcal{C}$ dans le plan de la facette de cône associée $\mathcal{T}_a$
  trianguler le polygone plan associé à ces contours.
fin pour

Algorithme 3 cherche_arêtes(V, E)

si arête de V à gauche de E manquante **alors**
  calculer la direction de recherche $\mathbf{l}_{gauche}$ de cette arête
  cherche_sommet($V, \mathbf{l}_{gauche}$)
fin si
si arête de V à droite de E manquante **alors**
  calculer la direction de recherche $\mathbf{l}_{droite}$ de cette arête
  cherche_sommet($V, \mathbf{l}_{droite}$)
fin si

Algorithme 4 cherche_sommet(V, $\mathbf{l}$)

calculer arête maximale E_{max} dans la direction de recherche $\mathbf{l}$
chercher l'image i telle que $cone_i$ intersecte E_{max} au plus près de V
si i existe **alors**
  chercher le point triple $P = E_{max} \cap cone_i$ dans la base de ceux déjà calculés
  si P n'est pas déjà créé **alors**
    calculer le point triple $P = E_{max} \cap cone_i$
    ajouter P au maillage et dans la base des points triples
  fin si
  ajouter la nouvelle arête E=[V, P] au maillage
  cherche_arêtes(P, E)
sinon
  chercher le sommet S de segment de vue incident à E_{max}
  ajouter la nouvelle arête [V, S] au maillage
fin si

3.9 Analyse de l'algorithme

3.9.1 Précision

Nous faisons ici quelques remarques sur la précision de l'algorithme. Nous avons fait remarqué au chapitre 2 qu'il était possible d'obtenir une discrétisation des silhouettes, garantissant que l'erreur de position de tout point de cette silhouette vectorisée est bornée par la taille du pixel. Tous les traitements en aval préservent cette information : c'est pourquoi l'erreur de tout point sur la surface finale calculée est bornée par rapport à une enveloppe visuelle idéale, si l'erreur de calibrage est elle aussi bornée. De plus cette borne est calculable en fonction des bornes d'erreurs du calibrage, de la taille d'un pixel, et de la taille de la scène, comprenant les caméras. Expérimentalement, dans les système usuels utilisés, avec un ordre de grandeur de l'espace d'acquisition de 10m et des caméras de résolution 640×480 ou plus, et des algorithmes de calibrage standard [54], nous constatons que cette borne d'erreur est de l'ordre du centimètre dans l'espace de visibilité commun des caméras. Ainsi l'algorithme proposé permet de maintenir bornée l'erreur dans la chaîne de traitement, et de garantir une très bonne précision pour l'enveloppe visuelle polyédrique calculée.

3.9.2 Complexité

Nous donnons ici les différents éléments concernant la complexité de l'algorithme final, en fonction de n, le nombre d'images, m, le nombre d'objets présents dans la scène réelle, et q, le nombre maximal de sommets par contour occultant polygonal utilisé. L'analyse de la première étape, le calcul des segments de vue, a été analysée au paragraphe 3.4.2. La deuxième étape (le calcul des arêtes d'intersection de cônes) et la troisième étape de l'algorithme (l'identification des facettes) sont analysées ici.

Nombre de points triples L'étape de calcul des segments de vue calcule $O(nm^2q)$ segments de vues, sous l'hypothèse de convexité des m objets considérés. Le nombre de points triples est du même ordre de grandeur : tout comme les segments de vue, il s'agit d'éléments géométriques constitutifs des bandes de l'enveloppe visuelle apparaissant à l'incidence entre bandes. Sur chacun des $O(m^2)$ objets potentiellement reconstruits, $O(n)$ bandes apparaissent, chacune des bandes ayant $O(q)$ primitives. Il y a donc $O(nm^2q)$ point triples et arêtes d'intersection de cône sur toute enveloppe visuelle polyédrique. Le temps de calcul de l'identification des facettes, est trivialement en $O(nm^2q)$, de l'ordre de grandeur du nombre de primitives du polyèdre.

Nombre d'opérations par arête de recherche En vue de trouver la complexité totale de la reconstruction des arêtes d'intersection de cônes, il nous faut maintenant trouver le coût de construction pour chacune de ces arêtes. Clairement, d'après l'algorithme, la recherche de points triples potentiels est la tâche largement dominante pour chaque arête, car elle fait intervenir les cônes de vue de toutes les autres images. Deux autres termes sont présents dans l'expression, le premier pour la recherche d'un sommet de segment de vue lorsqu'aucun point triple n'est trouvé ($O(\log m)$ car il y a $O(m)$ primitives sur une ligne de vue, du fait de l'hypothèse de convexité) et le second pour la recherche de l'existence d'un point triple déjà calculé, dans une structure d'indexation. On ne traite qu'un graphe de points triples à la fois, ce qui permet de limiter la taille de la structure de recherche, puisque l'on cherche un ensemble de points triples connexes, différent pour chaque graphe. Le nombre de points triples dans un même graphe connexe est en $O(nq)$, puisqu'il fait intervenir les primitives d'un seul des m^2 objets potentiellement reconstruits à la fois. La recherche de points triples est donc en $O(nq \log nq)$.

Concentrons-nous donc sur le terme dominant : la recherche du sommet implique d'examiner dans toutes les images toutes les facettes de cônes visuels (et donc de manière équivalente toutes les arêtes des silhouettes polygonales dans les images) pouvant engendrer un point triple par intersection avec la direction de recherche. Une solution naïve peut donc être implémentée en $O(nmq)$ pour chaque arête, qui engendrerait une complexité de $O(n^2m^3q^2)$ pour le calcul de toutes les arêtes.

Tout comme dans le cas des segments de vue cependant, il n'est pas nécessaire d'effectuer une recherche exhaustive dans les images. Une décomposition appropriée de l'espace image (quad-tree, kd-arbre, ou BSP) permet de réduire le coût de recherche de la première intersection dans les images à $O(n \log mq)$ par recherche des plus proches arêtes voisines autour de la projection de l'arête maximale de recherche dans chaque image. Une telle décomposition entraîne l'ajout d'un coup global de $O(nmq \log mq)$ opérations en tant que précalcul. Notons que la décomposition requise ici est une construction indépendante pour chaque vue, contrairement à la rectification épipolaire utilisée pour les segments de vue.

Nombre total d'opérations Etant donné qu'une recherche en $O(n \log mq)$ est a effectuer pour chacune des $O(nm^2q)$ arêtes d'intersection de cône, le calcul de toute arête nécessite $O(n^2m^2q \log mq)$ opérations. Ainsi cette complexité domine légèrement celle du calcul des segments de vue, étant quadratique avec le nombre d'objets dans la scène. Il s'agit donc de la complexité dominante pour le calcul d'une enveloppe visuelle polyédrique. Cette complexité est constituée de termes liés à l'initialisation : $O(n^2mq \log mq)$ pour la rectification épipolaire en vue du calcul des segments de vue, $O(nmq \log mq)$ pour la structure de recherche

d'intersection de cônes dans les images. Les termes dominant sont liés au calcul des segments de vue $O(n^2mq \log mq)$, et au calcul des intersections de cône en $O(n^2m^2q \log mq)$. Enfin l'identification des facettes du polyèdre est en $O(nm^2q)$, de l'ordre de grandeur du nombre de primitives du polyèdre.

3.9.3 Résultats

Nous avons implémenté l'algorithme présenté dans ce chapitre, et effectué un certain nombre d'expériences à titre de validation. Ces expériences ont été menées aussi bien sur des objets synthétiques (dont on créé les silhouettes par rendu d'image de synthèse) que sur des objets réels, dont les silhouettes ont été obtenues par acquisition. Pour ces dernières nous utilisons le plus souvent la plateforme multi-caméra de l'INRIA Rhône-Alpes, GrImage, que nous avons contribué à mettre en place, avec plusieurs équipes de recherche, et dont une description plus complète est donnée au chapitre 5.

Quelques résultats synthétiques permettent de se rendre compte de la capacité de reconstruction d'un algorithme basé sur l'enveloppe visuelle : un objet de topologie complexe, l'objet "noeuds", est reconstruit à partir de 42 vues (figure 3.24). Il permet d'illustrer le fait que l'enveloppe visuelle rend compte de toute l'information d'un objet visible à partir de ses silhouettes. L'objet noeuds comporte un grand nombre de parties *hyperboliques*, parties non convexes mais visibles à partir d'un point de vue. C'est aussi l'occasion d'illustrer les limites intrinsèques des approches classiques purement volumétriques, en comparant visuellement la reconstruction obtenue par une telle méthode. La précision offerte par notre méthode est très supérieure à celle obtenue sur une telle grille, du fait de la discrétisation de la surface. Une précision équivalente ne peut être obtenue qu'en employant un très grand nombre de voxels.

Un deuxième résultat visuel montre la mise en pratique de l'algorithme dans un contexte d'images réelles. Il est ainsi possible d'obtenir, pour quatre vues, une enveloppe visuelle polyédrique en temps réel. L'objet est reconnaissable, mais la précision vis à vis de la surface originale de l'objet est médiocre. Il est nécessaire d'utiliser un plus grand nombre de vues pour obtenir une meilleure précision. Pour avoir une meilleure idée du rapport entre précision et nombre de vues utilisées, nous présentons en figure 3.26 des reconstructions d'un tore obtenues avec un nombre de vues croissant. Pour un nombre faible de vues, nous constatons à nouveau que la surface initiale et la reconstruction ont des genres différents : le polyèdre de l'enveloppe visuelle est percé de deux trous dans ce cas, à cause d'une ambiguïté visuelle. Aucune vue ne permet d'éliminer la branche fantômes apparaissant devant le tore. A l'inverse, la reconstruction avec 42 points de vues est plus précise mais est constituée d'un grand nombre de petites primitives : un grand nombre de vues contribuent à de petits endroits. Nous arrivons ici aux limites de la précision sur la surface de l'enveloppe visuelle : rajouter des vues

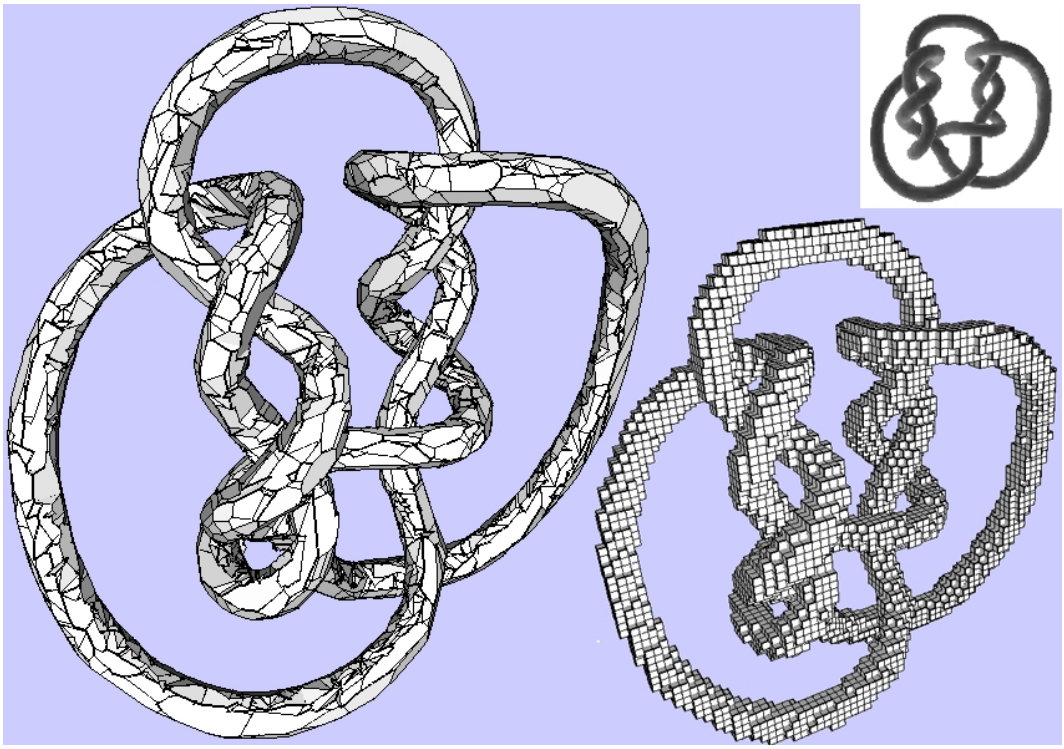


FIG. 3.24 – Enveloppe visuelle de l'objet "noeuds" (tiré de la page web de Hoppe [52]), obtenue avec notre méthode en utilisant 42 points de vues autour de l'objet. La reconstruction de ses 19528 sommets prend 5.53 secondes sur un PC 2.4GHz et 1Go de RAM. La discrétisation des silhouettes engendre des polygones d'environ 200 sommets en moyenne. La reconstruction de la même forme sur une grille de voxels est beaucoup moins précise, en utilisant une grille bien ajustée de $64^3 = 262144$ voxels. En haut à droite, une vue du modèle original.

supplémentaires ne permettra pas d'obtenir une meilleure précision, la taille des primitives ajoutées sur la surface étant de l'ordre de grandeur de la borne d'erreur de reconstruction. Il est donc rarement utile, dans le contexte de notre problème de vision, de considérer plus de quelques dizaines de vues. En pratique, l'utilisation de 10 à 20 vues offre généralement un bon rapport entre précision et vitesse de calcul.

Nous donnons ci-dessous des données plus détaillées sur le temps de calcul de l'algorithme.

Implémentation

Nous avons implémenté deux versions de l'algorithme, l'une non optimisée (en $O(n^2m^2q^2)$) d'une mise en oeuvre plus simple, et l'autre ayant fait l'objet d'op-

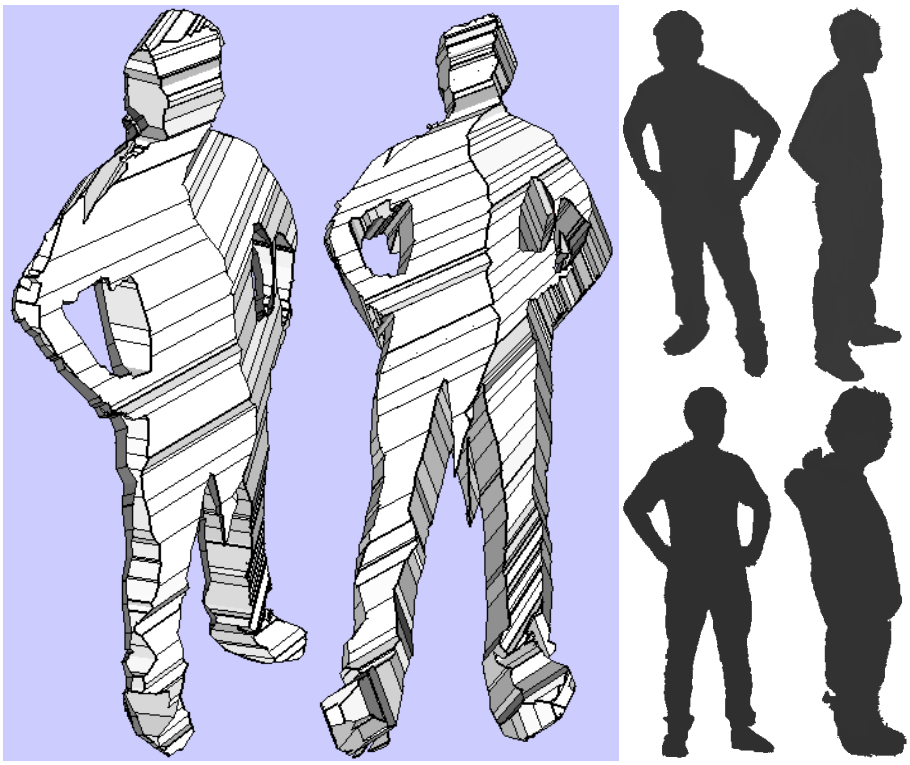


FIG. 3.25 – Deux rendus de l'enveloppe visuelle d'une figure humaine, obtenus grâce aux silhouettes acquises de quatre points de vue (montrées à droite). Image acquises de résolution 640×480. 2512 sommets calculés en 80msec, avec des silhouettes à 263 sommets.

timisations pour la recherche des arêtes d'intersection de cônes sur le polyèdre (utilisée pour les mesures de performances). En pratique cette recherche peut être très efficace, selon la décomposition de l'espace image utilisée. Si celle-ci est bien choisie l'algorithme peut se limiter à une recherche dans une zone très locale, autour de la projection de l'arête maximale de recherche. On préférera donc à ce titre une décomposition selon les deux axes dans les images, comme une grille, ou un kd-arbre, par exemple. Nous avons obtenu une implémentation très efficace en utilisant une décomposition de l'image sur une grille régulière. L'espace image se trouve divisé en cellules carrées où l'on stocke l'ensemble des arêtes que cette cellule contient ou intersecte. Cette décomposition ne garantit pas toujours le comportement logarithmique de l'algorithme, que lui conférerait un kd-arbre par exemple. Néanmoins l'implémentation fournie s'est déjà avérée très rapide dans les conditions d'utilisation attendues dans un contexte de reconstruction. Celles-ci font rarement intervenir plus de quelques dizaines d'images, avec des silhouettes polygonales comportant quelques dizaines ou centaines de sommets. La valeur de m est souvent faible et généralement très inférieure à q. Au vu des temps

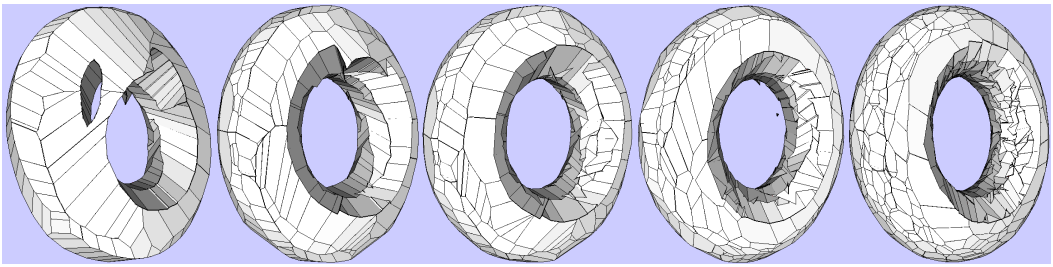


FIG. 3.26 – Enveloppes visuelles d'un tore obtenues avec un nombre de vues croissant, avec des silhouettes à 60 sommets.

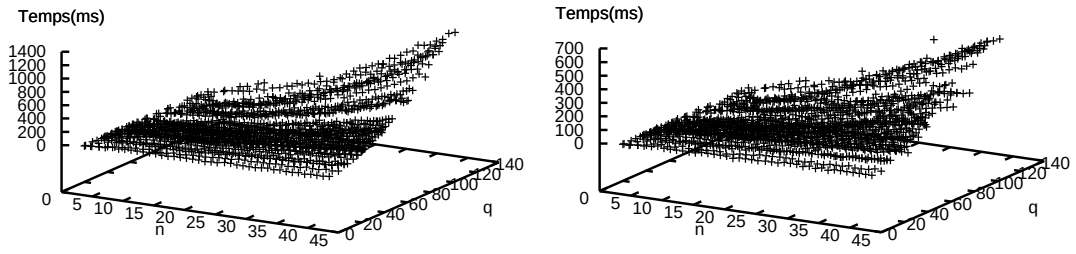


FIG. 3.27 – Comportement de l'algorithme sur un échantillon de 1760 reconstructions, en fonction du nombre d'image n et du nombre de sommets dans les silhouettes polygonales q. A gauche, le temps de calcul des segments de vue, à droite le temps de calcul des arêtes d'intersection de cônes.

d'exécutions très faibles obtenus, nous n'avons pas poussé les optimisations beaucoup plus complexes permettant d'atteindre le meilleur comportement théorique de l'algorithme, dont le bénéfice ne pourrait se manifester que pour des ordres de grandeur de n et q beaucoup plus grands et ayant peu d'intérêt pratique.

Le comportement de notre implémentation et des principales sous-tâches le composant est illustré par les courbes en figure 3.28 et 3.27. La tâche de facétisation est pratiquement négligeable (moins de 3% du temps de calcul global).

3.9.4 Comparaisons

Comme nous l'avons vu en introduction du chapitre 3, il existe diverses approches pour calculer l'enveloppe visuelle. Nous avons comparé qualitativement l'approche avec les méthodes volumiques, où des voxels sur une grille cubique sont sculptés en fonction de leur appartenance à l'enveloppe visuelle. Certes le rapport entre la qualité du modèle produit et le temps passé à calculer la représentation est clairement en faveur de notre méthode ; celle-ci offre en outre de meilleures garanties en terme d'erreur de reprojection sur les silhouettes de départ, au vu de la chaîne de traitement qui permet de borner l'erreur. Cependant notre méthode

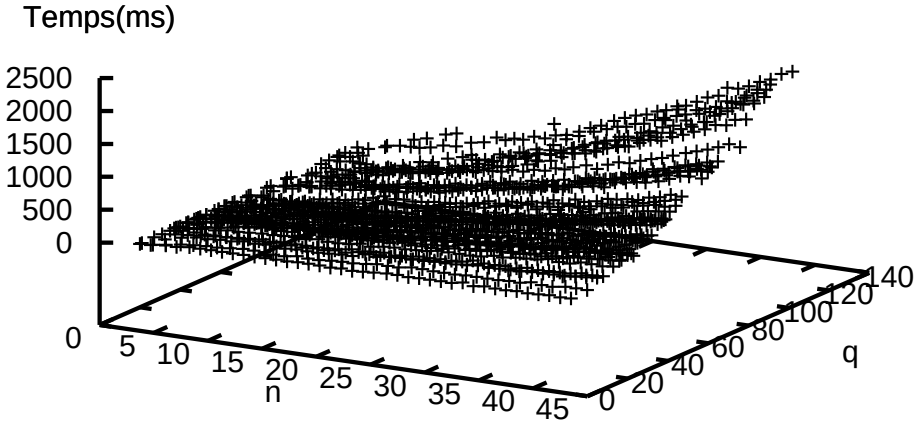


FIG. 3.28 – Temps de calcul global en fonction de n et q.

est une approche très différente des approches voxéliques, et l'une ou l'autre peut être choisie en fonction des besoins spécifiques de l'application visée. Pousser davantage la comparaison entre les deux méthodes n'aurait donc pas beaucoup de sens.

En revanche, nous avons aussi vu qu'il existait un certain nombre d'algorithmes pouvant être utilisés pour calculer une représentation très similaire ou identique de l'enveloppe visuelle, sous forme d'un polyèdre d'intersection de cônes visuels (sous réserve de prendre la même polygonalisation des silhouettes dans les images). Nous allons donc nous positionner plus finement par rapport à ces méthodes.

Rappelons que la méthode la plus efficace à ce jour proposée en vision pour calculer des enveloppes visuelles est celle de Matusik *et al.* [73], qui calcule des bandes discrètes et non pas directement un polyèdre. Un travail fondateur en terme d'enveloppes visuelles polyédriques est celui de Baumgart [11] ; néanmoins l'algorithme qu'il propose est un des ancêtres d'une deuxième catégorie d'algorithme d'intersections de polyèdres, les algorithmes de CSG. En outre, l'algorithme de Baumgart, basé sur des intersections deux à deux de polyèdres de forme générale, possède donc une complexité comparable aux méthodes de cette famille. Nous avons choisi, à titre de comparaison pour ces méthodes, la librairie CGAL [2]. Cette librairie utilise une représentation spéciale des polyèdres, les polyèdres Nef, très adaptés au calcul d'opérations booléennes, et propose des optimisations issus de travaux récents [49]. On trouvera notamment dans cette dernière publication une estimation de la complexité asymptotique du temps de calcul de l'algorithme pour l'intersection de deux polyèdres, en $O((F1 + F2 + S)\log(F1 + F2))$, où S est le nombre de primitives supplémentaires créées lors de l'intersection, et $F1$, $F2$ le nombre de facettes de chacun des polyèdres. Cette complexité est bien

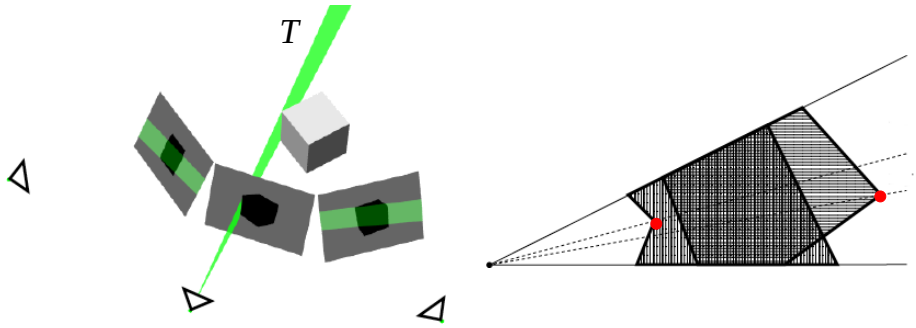


FIG. 3.29 – Méthode de Matusik *et al.* (tiré de [73]). (gauche) Calcul d'intersections polygonales avec les silhouettes pour la projection d'une face triangulaire infinie $\mathcal{T}$. (droite) Calcul de la contribution des n silhouettes par intersection dans le plan de $\mathcal{T}$. Les points marqués en rouge sont calculés et stockés par cet algorithme, mais il ne sont jamais considérés par notre méthode, d'où le gain en complexité.

représentative de l'ensemble des algorithmes de calculs d'intersection de polyèdres de type CSG. A notre connaissance, aucun d'entre eux ne fait mieux.

Nous axerons donc notre comparaison autour des deux méthodes représentant le mieux l'état de l'art, celle de Matusik *et al.* [73] et celle de CGAL [49]. Étant donné que l'aspect de la surface calculé est identique (même si [73] ne propose pas de solution satisfaisante pour connecter les bandes calculées et garantir la fermeture et l'absence d'auto-intersection de la surface) nous allons nous intéresser à ce qui différencie plus nettement les méthodes proposées : la complexité asymptotique de leur temps de calcul.

Méthode de Matusik *et al.*

En ce qui concerne la méthode [73], l'analyse nous est fournie par les auteurs eux-mêmes dans un rapport de recherche [75]. Rappelons à cet effet que l'algorithme proposé par Matusik *et al.* calcule la contribution à l'enveloppe visuelle dans le plan d'une face triangulaire infinie $\mathcal{T}$. Deux étapes sont nécessaires. Premièrement il faut calculer l'intersection de la projection de cette face infinie et les silhouettes des $n-1$ autres images (intersections ne faisant donc intervenir chacune que deux images). Puis il faut ramener chacun des polygones trouvés dans le plan de la facette par homographie pour ensuite les intersecter entre eux (voir figure 3.29). Ceci permet de calculer toutes les facettes du polyèdre de l'enveloppe visuelle dans le plan de $\mathcal{T}$.

Le temps de calcul est en $O(n^2mq \log mq + nl(nmq + b) \log nmq)$ où b est le nombre d'intersections se produisant entre toutes les facettes dans un même plan, et où l est le nombre maximal d'intersections d'une ligne de vue avec un cône

visuel, selon la propre définition des auteurs. En se ramenant aux conditions de notre analyse, nous pouvons identifier l à m, en supposant que l'on manipule des régions convexes. Le premier terme concerne le calcul des intersections à deux images, le deuxième concerne le calcul lui-même et est le terme prépondérant. C'est donc ce terme, $nm(nmq + b) \log nmq$, qu'il faut mettre en relation avec la complexité de notre méthode, $O(n^2m^2q \log mq)$. En pratique b est d'un ordre de grandeur comparable à $O(nmq)$. Le nombre d'opérations de la méthode est alors en $O(n^2m^2q \log nmq)$, moins bon que celui affiché par notre algorithme. La différence s'explique notamment par le fait que l'algorithme calcule et stocke une représentation intermédiaire, les $n - 1$ polygones d'intersection avec une silhouette. Ces représentations intermédiaires comprennent des primitives potentiellement inutiles pour le polyèdre final. Mais c'est seulement l'étape suivante, la fusion entre tous ces polygones par intersection, qui permettra d'extraire l'information pertinente de cette représentation intermédiaire. Une telle étape intermédiaire est inexistante dans notre algorithme, où chaque calcul contribue directement à la création d'une primitive du polyèdre final.

Méthodes CSG

Les méthodes CSG et CGAL affichent, comme mentionné plus tôt, une complexité dominante en $O((F1 + F2 + S) \log(F1 + F2))$. La caractéristique de ces algorithmes est qu'il sont conçus pour le calcul d'opération booléennes entre deux polyèdres de forme générale. A notre connaissance, aucun algorithme n'a été proposé pour calculer directement l'intersection de n polyèdres dans le cadre des CSG. Il faut donc utiliser successivement les intersections deux à deux pour obtenir une enveloppe visuelle, en partant de cônes visuels tronqués en tant que polyèdres initiaux. Étant donné que le nombre de primitives sur l'enveloppe visuelle est proportionnel aux nombre de vues utilisées et au nombre de sommets dans les contours images ($O(nm^2q)$), la plus faible complexité pour effectuer les n intersections de cônes est obtenue en réalisant les intersections de manière arborescente, de sorte que toute intersection réalisée fasse intervenir des polyèdres du même ordre de grandeur[3]. Le nombre total d'opérations attendu pour reconstruire le polyèdre de l'enveloppe visuelle à partir de cônes de vue est alors de $O(nm^2q \log n \log nm^2q)$.

Ce résultat théorique est très intéressant, mais se doit d'être relativisé. L'intérêt du résultat est qu'il suggère que l'algorithme que nous proposons n'est pas encore optimal dans sa version présentée et peut être amélioré. Il doit être

[3]une discussion avec Peter Hachenberger suggère l'implémentation optimale sous forme de file de priorité, où l'intersection est réalisée à tout moment sur les deux polyèdres de taille minimale et remise dans la file, en bouclant jusqu'à obtention du résultat.

relativisé car en pratique plusieurs facteurs jouent en défaveur des algorithmes de type CSG.

Le $\log(F1 + F2)$ des intersections deux à deux ne peut être obtenu qu'en construisant une structure permettant la recherche des plus proches voisins dans l'espace $\mathbb{R}^3$, comme un kd-arbre construit sur les axes d'un repère choisi de $\mathbb{R}^3$ (ce qui est le cas dans CGAL). Ce précalcul est réeffectué pour chaque polyèdre intermédiaire obtenu, engendrant un surcoût total important à l'algorithme. Même s'il n'en change pas la complexité théorique, il la pénalise d'une constante multiplicative importante. Egalement, les algorithmes CSG ont une vocation plus généraliste et sont donc souvent alourdis par des vérifications et prérequis inutiles dans notre cas de figure. A cause des divers surcoûts, le bénéfice de cette meilleure complexité théorique ne peut être obtenu que pour des valeurs de n et q très grandes. Ceci est d'autant plus vrai pour CGAL, qui est conçue pour fournir des résultats très fiables, au détriment de la vitesse de calcul. En effet cette librairie repose sur des calculs à précision arbitraire pour une évaluation exacte du signe de déterminants, ce sans quoi l'algorithme ne peut pas fonctionner. Les algorithmes réalisant la CSG peuvent en outre se heurter à la spécificité du problème, qui conduit la plupart des implémentations testées à une terminaison pure et simple à cause de l'incapacité à gérer les proches cotangeances, inhérentes à l'enveloppe visuelle.

Les résultats pratiques corroborent ces allégations. Ils nous a été très difficile de trouver une librairie qui fonctionne pour effectuer les comparaisons. Parmi les trois librairies Open Source essayées, CGAL[2], GTS[4], b-rep library[1], seule CGAL fournit des résultats exploitables. Parmi ces librairies, CGAL est la seule à faire appel à des évaluations exactes de signes de déterminants. Il ne nous a pas été possible d'obtenir une seule intersection de cônes correcte avec gts ou b-rep lib. Il nous a de plus été rapporté par John Isidoro, étudiant à Boston University cherchant une librairie de CSG pour reconstruire des enveloppes visuelles, qu'une quatrième librairie de CSG, TWIN [7], présentait des problèmes similaires et des taux de fiabilité faibles. Il ne nous a pas été possible de tester cette librairie, qui n'est pas open source, et disponible que pour quelques plates-formes spécifiques.

3.9.5 Fiabilité

Il est important, pour nombre d'applications, d'avoir un algorithme de reconstruction fiable. Nous avons mentionné la difficulté d'une telle tâche, au paragraphe 3.1.3. La nature géométrique de l'algorithme et l'existence de situations dégénérées peuvent le mettre en difficulté.

Nous donnons ici quelques données sur la fiabilité de l'algorithme proposé. L'implémentation que nous en avons fait effectue un certain nombre de vérifications structurelles au cours de l'exécution. Elle est donc capable de détecter les problèmes rencontrés lors de la création de la surface. Ces erreurs arrivent

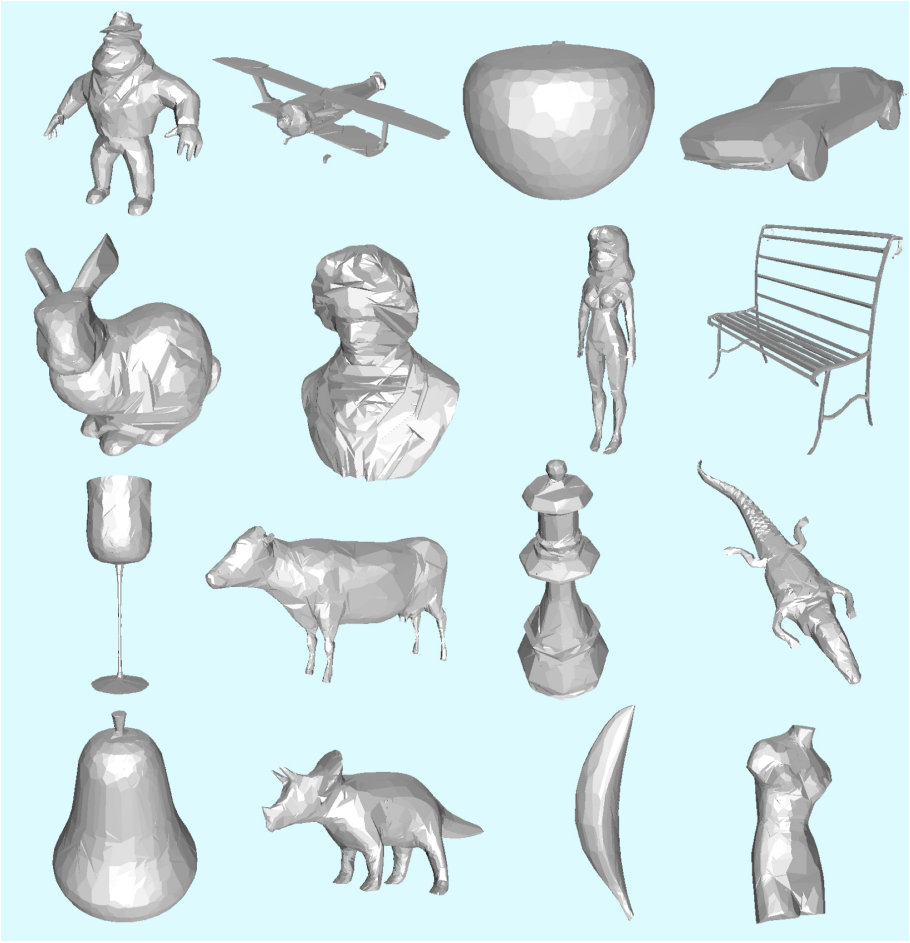


FIG. 3.30 – Quelques modèles reconstruits avec l'algorithme proposé, avec 42 vues.

lorsque l'on se rapproche d'une situation non générique. Bien que ce cas se produise très peu souvent, il se peut alors qu'une intersection ne soit pas détectée par l'algorithme bien que réellement présente sur la surface. Dans une telle situation l'arête en cours de recherche est abandonnée, créant potentiellement un trou dans la surface. Notre implémentation détecte ces incidents et les énumère. L'algorithme possède cependant un avantage contre ce type de problèmes : outre le fait que ces situations sont rendues statistiquement très improbables en perturbant numériquement les sommets des contours d'entrée, l'algorithme est capable de se rattraper en cas d'erreur. En effet une arête peut être visité de quatre manières, puisque l'on peut explorer une arête en venant de n'importe laquelle des quatre arêtes lui étant incidentes. Un incident peut ainsi être contourné, et réparé par un parcours de l'algorithme passant par une de ces arêtes, où l'ambiguïté numérique génératrice d'incident ne se manifeste plus.

Pour vérifier de manière pratique la fiabilité de notre algorithme, nous avons ef-

fectué une série de calculs d'enveloppe visuelle sur 45 objets synthétiques, pour 3 à 42 vues de celui-ci, c'est à dire un total de 1760 reconstructions dans des conditions très disparates. Un échantillon des modèles obtenus est donné en figure 3.30. Ces expériences ont été menées pour les deux versions de notre implémentation (optimisée et non optimisée). Le tableau suivant fait état du pourcentage d'expériences où des incidents se sont produits, ainsi que le pourcentage de reconstructions comportant réellement des erreurs et produisant une surface non fermée. Les surfaces présentant réellement des erreurs sont identifiées par un test de cohérence des structures a posteriori.

	Taux d'incidents	Taux d'erreur
Version optimisée	4.83%	2.61%
Version sans optimisation	2.10%	0.06%

TAB. 3.1 – Taux d'incidents et d'erreurs de reconstruction sur un total de 1760 échantillons.

Il est remarquable de constater que pour la version simple (et purement quadratique) de l'algorithme, sur 1760 reconstructions, seule une a véritablement échouée (0.06%). La quasi-totalité des cas comportant des incidents de reconstruction, déjà peu nombreux, a été rattrapée. Les deux versions devraient produire des résultats identiques : les erreurs de la version optimisée sont dûs à des défauts d'implémentation de la subdivision dans les images, qui est complexe à mettre en oeuvre. Ces résultats permettent néanmoins à l'algorithme que nous proposons de rivaliser avec l'implémentation de CGAL réputée 100% robuste, et ce malgré l'utilisation de types à précision fixe (double). De plus la solution que nous proposons est plus rapide que CGAL de plusieurs ordres de grandeurs pratiques, dans des conditions d'utilisations réelles : pour une reconstruction à quatre poins de vue d'une forme humaine, le temps d'exécution de CGAL se chiffre en minutes. En outre le taux de succès de 100% d'une librairie de CSG exacte comme CGAL se fait au prix d'une mise en œuvre lourde : il faut en effet construire spécialement des cônes de vue polyédriques préalablement à toute intersection. Dans les faits cela signifie la construction de portions de cônes de vue tronqués à l'échelle de la scène qui soient conforme aux conditions attendues par CGAL, sous peine de rejet. Ceci nécessite des ajustements manuels pour chaque nouvelle scène. La solution spécifique que nous proposons pour le calcul du polyèdre de l'enveloppe visuelle est entièrement automatique et présente une fiabilité comparable. Nous avons donc montré qu'il était possible de concilier rapidité et fiabilité au sein d'un seul et même algorithme.

3.10 Conclusion

Nous avons proposé, dans ce chapitre, une méthode permettant de modéliser la surface d'un objet à partir de ses silhouettes. En exploitant la nature discrète du problème et le fait que des silhouettes polygonales parfaitement fidèles peuvent être obtenues dans les images, nous avons montré qu'il était possible de propager efficacement cette information pour calculer une enveloppe visuelle polyédrique rendant compte d'une caractérisation binaire des silhouettes en chaque pixel des images. L'algorithme utilise les segments de vue en tant que représentation initiale discrète des bandes de l'enveloppes visuelles, et reconstitue les primitives manquantes de proche en proche. Pour ce faire nous avons étudié la structure du polyèdre, ce qui nous permet de comprendre comment construire les arêtes d'intersection de cônes initialement absentes de la représentation. Nous donnons des résultats théoriques et numériques sur le comportement attendu de l'algorithme, aussi bien en terme du nombre de primitives que l'on peut s'attendre à trouver sur le polyèdre, qu'en estimant la complexité asymptotique de son temps de calcul.

En outre la méthode est comparée à l'état de l'art, y compris les approches de géométrie algorithmique qui peuvent permettre de résoudre le problème, et qui n'ont pas fait l'objet de comparaison rigoureuse auparavant pour le problème posé. Il a été montré que l'algorithme surpasse l'état de l'art en performances pures et rivalise avec les approches exactes à précision arbitraire en terme de fiabilité, en produisant des polyèdres à la topologie d'une variété de manière répétable. L'algorithme proposé constitue donc une approche nouvelle rigoureuse au problème de reconstruction d'enveloppe visuelles à partir de silhouettes binaires.

Grâce à cet algorithme, il est ainsi possible de calculer une enveloppe visuelle d'erreur bornée dès que l'on a accès à une information de silhouette très fiable. Comme la pratique le montre cependant, il est malheureusement difficile d'extraire des silhouettes binaires de manière rapide et automatique, à moins d'utiliser un environnement très contraint (fond bleu, studio). Néanmoins, nous avons accès à une information plus nuancée en chaque image et qui est sous exploitée, permettant d'exprimer la probabilité de trouver une silhouette sachant la couleur. Nous explorons, dans le chapitre suivant, une solution possible pour exploiter cette information et rendre l'estimation de forme à partir de silhouettes plus robuste au bruit dans les images.

4.

Reconstruction statistique

Nous avons montré qu'il était possible de résoudre le problème de modélisation à partir de silhouettes de manière purement géométrique, dès que l'on dispose de silhouettes binaires fiables. Une telle caractérisation est cependant difficile à obtenir de manière automatique : en pratique la segmentation des silhouettes, basée sur la couleur des objets dans les images, est sujet à diverses formes de bruit qui en perturbent l'estimation.

Plutôt que de fixer prématurément un seuil de croyance dans chaque image pour extraire notre silhouette d'intérêt, et de n'utiliser qu'un seul instant de la séquence d'images, comme il est souvent fait en pratique, nous souhaitons fusionner les différentes sources d'information provenant de toutes les images, et de différents instants. L'idée est de se rapprocher d'un paradigme de soustraction de fond multi-vue, où la redondance d'information, et la confiance qu'on lui accorde, participent à une estimation plus robuste de la scène.

Nous proposons alors une nouvelle expression de notre problème d'estimation de forme, sous forme de fusion Bayésienne de capteurs, pour intégrer ces informations. Pour pouvoir exprimer la fusion Bayésienne dans l'espace, nous optons pour une représentation de la scène sous forme d'une grille de probabilités d'occupation. Nous associons au problème un modèle capteur complet pour chaque pixel, rendant compte de la chaîne complète d'incertitudes influençant la formation d'une observation dans les images. Nous montrons les bénéfices d'une telle modélisation pour une scène statique, avant de montrer les possibilités d'extension du système permettant de prendre en compte la dynamique de la scène.

4.1 Introduction

Les méthodes de reconstruction géométriques, telles celles précédemment proposées, partent souvent d'un étiquetage des pixels en deux catégories, le fond et l'avant-plan, pour identifier la silhouette des objets. Cet étiquetage est généralement réalisé séparément dans chaque image, avant tout calcul en 3D. Malheureusement, un tel étiquetage monoculaire est difficile à réaliser de façon fiable dans un environnement quelconque et peu contrôlé. Diverses perturbations expliquent cette difficulté : le bruit des capteurs CCD, les ambiguïtés de couleur entre le fond et l'avant-plan, les changements d'illumination de la scène (ce qui inclut les ombres d'objets d'intérêt), etc. De plus, des erreurs dans un tel processus peuvent avoir des conséquences drastiques sur la perception 3D multi-caméra, particulièrement en présence de bruit de calibrage des caméras, ou si l'acquisition des images présente des défauts de synchronisation entre les différentes vues.

Notre but est donc de trouver une représentation de l'information silhouette multi-vue, pour laquelle l'inférence sur les silhouettes est plus robuste aux différentes sources d'incertitude mentionnées ci-dessus. Intuitivement, la connaissance simultanée de toutes les images nous apporte plus d'information sur les silhouettes que la connaissance d'une seule de ces images. Cette idée nous a conduits à calculer une fusion Bayésienne de l'information silhouette en 3D, pour prendre en compte de manière optimale la contribution de chaque image. Le résultat d'une telle fusion contient naturellement une information de forme, et peut donc être utilisée pour des applications de modélisation classique à partir d'images ; mais elle peut aussi être utilisée pour améliorer l'extraction des silhouettes dans les images, pour toute application basée sur les silhouettes.

Les méthodes existantes utilisent généralement deux tâches distinctes pour inférer la forme des objets à partir de silhouettes : une décision sur la localisation des silhouettes est d'abord effectuée pour chaque image, puis la forme des objets est calculée géométriquement à partir des silhouettes grâce aux techniques liées à l'enveloppe visuelle. Bien que le calcul d'enveloppe visuelle puisse être exact à un jeu de silhouettes donné comme nous l'avons montré, le modèle produit souffre d'une décision binaire sur chacune de ces silhouettes, héritant des défauts dûs aux perturbations précédemment mentionnées. Nous proposons ici une approche qui permet de repousser la décision d'occupation, et donc une meilleure intégration de l'information disponible.

Parmi les travaux les plus proches de notre problématique, Goldlücke *et al.* [45] proposent une estimation simultanée de la disparité stéréo et des silhouettes, avec cependant le coût élevé de l'optimisation globale pour garantir la robustesse. Zeng *et al.* proposent une soustraction de fond multi-vue, avec un schéma itératif assez coûteux et dont la convergence n'est pas assurée, avec la contrainte supplémentaire de la visibilité de l'objet d'intérêt par toutes les caméras [108].

La prise en compte simultanée d'informations provenant de plusieurs silhouettes a déjà fait l'objet d'une méthode [94], avec cependant une modélisation plus rudimentaire de la formation de silhouettes et une formulation discrète de l'occupation résolue avec une coupure de graphe, donnant des résultats moins précis. La satisfaction de contraintes liées aux silhouettes dans plusieurs images a aussi été utilisée dans un contexte d'optimisation par méthode variationnelle, avec l'idée de produire une surface dont les contours occultants passent aux endroits de forts gradients dans l'image [60]. Des travaux ont été effectués en robotique pour localiser un objet à partir d'une séquence d'images acquises à partir d'un robot [71], avec cependant une problématique et une mise en œuvre différente de la nôtre. Plusieurs travaux ont aussi exploré la possibilité de représenter l'espace avec une grille de probabilité, pour résoudre d'autres problèmes tels la stéréo (wide-baseline) [19] ou la reconstruction d'objets transparents [13]. Grauman *et al.* [47] proposent une méthode intéressante pour estimer l'ensemble de silhouettes le plus probable d'un même objet, en réalisant un apprentissage *a priori* sur cet ensemble avec une base de silhouettes humaines, présentant l'avantage d'une meilleure intégration de l'information sémantique, mais avec une généricité et une précision limitée. Toutes ses approches résolvent des problèmes connexes, avec certaines limitations. Nous présentons une approche bas niveau que nous espérons plus générale, et nous proposons d'enrichir l'information silhouette 2D en réalisant sa fusion dans une représentation 3D, tout en restant générique par rapport aux applications.

Nous proposons un nouvel outil basé sur les grilles d'occupation [33] : il s'articule autour d'une représentation de la scène sous la forme d'une grille de voxels pour laquelle nous calculons la probabilité d'occupation d'un objet d'intérêt, associée à un modèle capteur. Les grilles d'occupation ont été beaucoup utilisées dans la communauté robotique [27], pour représenter et reconstituer l'environnement de navigation d'un robot à partir d'observations de sonars, mais aussi de données stéréo [78]. Notre contribution consiste à étendre le concept de grille d'occupation aux capteurs CCD, et à reformuler le problème d'estimation de forme à partir de silhouettes comme un problème de fusion de capteurs. Pour ce faire, nous associons à chaque pixel un modèle capteur *génératif*, où l'on modélise la réponse du pixel aux occupations de voxels dans la scène. Notre formulation tient compte de la région visible par chaque pixel, du problème d'échantillonnage des voxels, des petites erreurs de calibrage, et de la fiabilité du pixel en tant que capteur. Ce modèle est alors utilisé pour résoudre la question inverse plus difficile : savoir où se trouve la matière dans la scène, connaissant les couleurs observées par tous les pixels. Nous montrons aussi que la grille d'occupation calculée peut être utilisée pour réaliser une soustraction de fond multi-vue, où l'estimation de la silhouette dans chaque vue bénéficie de la connaissance acquise dans d'autres vues.

Ces travaux ont fait l'objet d'une publication et d'un rapport de recherche en collaboration avec Edmond Boyer [40, 41].

4.2 Formulation du problème

Nous considérons le problème de fusion des informations silhouette multi-vue. Nous supposons disposer d'un jeu d'images *courantes*, obtenues à partir de caméras complètement calibrées. Nous supposons également disposer d'un jeu d'images du fond, obtenues préalablement à partir de la scène dénuée d'objets d'intérêt. Aucune supposition n'est faite quant à l'existence d'une région de visibilité commune à toutes les caméras.

Nous formulons le problème comme l'estimation Bayésienne pour chaque voxel, de la probabilité avec laquelle celui-ci est occupé par un objet d'intérêt. Nous utilisons un modèle capteur génératif : nous considérons que l'occupation des voxels est la cause, et les observations les effets. Nous modélisons donc l'influence de chaque voxel sur la formation des images. Ceci nous permet de résoudre, grâce à l'inférence Bayésienne, le problème inverse : exprimer la vraisemblance de l'occupation des voxels à partir des images, traitées comme des mesures bruitées de la scène.

La résolution d'un problème bayésien requiert l'expression de la distribution de probabilité conjointe de co-ocurrence de toutes les variables du problème (définies au paragraphe 4.2.1), avant toute inférence. Cette distribution de probabilité conjointe doit ensuite être décomposée et simplifiée, en fonction des dépendances statistiques que l'on considère entre les variables (paragraphe 4.3). En particulier, des formes paramétriques doivent être affectées aux différents termes de la décomposition pour donner une forme explicite aux relations liants plusieurs variables (paragraphes 4.3.2 et 4.3.3). Ceci réduit considérablement la complexité des calculs effectués à partir de la distribution conjointe, lors de l'inférence avec la règle de Bayes, utilisée pour calculer les distributions des variables d'occupation des voxels (paragraphe 4.4).

4.2.1 Principales variables du problème

Nous dénotons l'ensemble des n images courantes $\mathcal{I}$. $\mathcal{I}^i$, $i = 1 \cdots n$, représente alors les données image de la caméra i, et $\mathcal{I}^i_p$ représente les données image au point p dans l'image i, exprimées dans un certain espace de couleur (RGB, YUV, etc). Bien que non étudié dans cette thèse, on peut imaginer sans perte de généralité avoir plus de données image, tel le gradient ou le Laplacien image par exemple, encapsulées dans le terme $\mathcal{I}^i_p$. Nous supposons que les données image correspondant aux m images du fond statique observées pour chaque caméra peuvent être résumées dans un seul modèle de fond $\mathcal{B}^i$, $i = 1 \cdots n$, représentant les pa-

ramètres d'un modèle statistique utilisé pour la régression. $\mathcal{B}^i$ peut par exemple être constitué d'un ensemble de modèles indépendant par pixel, comme nous le verrons plus loin. Le but d'une telle régression est habituellement de modéliser le bruit capteur. Tous les jeux de données image sont produits par n caméras dont on connaît les matrices de projection P^i.

τ symbolise l'information *a priori* que nous introduisons dans le modèle. Ceci inclut notre connaissance de la scène, ce que nous connaissons des caractéristiques des capteurs, nos connaissances générales du problème.

Soit $\mathcal{G}$ notre grille d'occupation. A chaque point X de l'espace, de coordonnées (x, y, z) dans cette grille, nous associons une variable d'occupation binaire $\mathcal{G}_X \in \{0, 1\}$, respectivement libre ou occupée. Nous faisons l'hypothèse d'indépendance statistique entre les occupations des voxels ce qui nous permet de calculer chaque occupation de voxel de manière indépendante. Ceci est une hypothèse couramment utilisée pour rendre le traitement de grilles d'occupation possible en robotique [33]. Les résultats montrent que l'estimation indépendante telle que nous la proposons, même si elle n'est pas aussi exhaustive qu'une recherche globale sur tout l'espace des configurations possible de la grille, permet d'obtenir une information très robuste et utilisable, pour un coût en calcul beaucoup plus raisonnable.

Nous avons défini les variables d'entrée et de sortie de notre problème. Nous définissons un jeu de variables cachées importantes dans les images, les cartes de détection des silhouettes $\mathcal{F}^i$, $i = 1 \cdots n$. Ces cartes définissent, pour chaque pixel p de l'image i, une variable binaire de détection de silhouette $\mathcal{F}^i_p$. $\mathcal{F}^i_p = 1$ si le capteur au pixel p de l'image i *témoigne* de la présence d'un objet sur sa ligne de vue. Nous insistons sur cette définition, car il y a une possibilité qu'il y ait effectivement un objet sur la ligne de vue du pixel p, mais que le capteur *se trompe* et ne rapporte pas cette information, en raison de causes internes ou externes (la modélisation des défaillances capteurs est discutée en détail au paragraphe 4.3.2). Ces cartes de détection représentent l'information silhouette dans notre modèle, que nous souhaitons marginaliser dans l'inférence.

4.3 Décomposition de la distribution conjointe

Notre but est d'inférer l'occupation $\mathcal{G}_X$ d'un voxel à la position X, sachant $\mathcal{I}$, $\mathcal{B}$, et τ. De ce fait, nous devons modéliser l'effet de $\mathcal{G}_X$ sur les observations. Pour modéliser les relations entre les variables du problème, nous devons calculer la distribution conjointe de probabilité de ces variables, $p(\mathcal{G}_X, \mathcal{I}, \mathcal{B}, \mathcal{F}, \tau)$. Nous proposons la décomposition suivante, basée sur les dépendances statistiques exprimées dans la figure 4.1 :

$$\begin{aligned} p(\mathcal{G}_X, \mathcal{I}, \mathcal{B}, \mathcal{F}, \tau) = {} & p(\tau)\; p(\mathcal{B}|\tau)\; p(\mathcal{G}_X|\tau) \\ & p(\mathcal{F}|\mathcal{G}_X, \tau)\; p(\mathcal{I}|\mathcal{F}, \mathcal{B}, \tau) \end{aligned}$$

- $p(\tau)$, $p(\mathcal{B}|\tau)$ sont les distributions *a priori* de notre jeu de paramètres, et des paramètres de la régression des images du fond, respectivement. Comme nous n'avons aucune raison *a priori* de favoriser l'une ou l'autre des valeurs de ces paramètres, ou un aspect particulier des images du fond, nous affectons une distribution uniforme à ces termes. De ce fait ils disparaissent de toute inférence.
- $p(\mathcal{G}_X|\tau)$ est la vraisemblance *a priori* de l'occupation, qui pourrait varier en fonction de X par exemple. Celle-ci est indépendante de toute autre variable sauf τ. Comme nous ne souhaitons favoriser aucune position particulière des voxels dans l'espace, et que c'est surtout la régularisation de la grille induite par les observations image qui nous intéresse dans cette thèse, nous affectons une distribution uniforme à ce terme et l'ignorons donc par la suite.
- $p(\mathcal{F}|\mathcal{G}_X, \tau)$ est le terme de vraisemblance des silhouettes. Les dépendances considérées reflètent le fait que l'occupation des voxels explique la détection des silhouettes dans les images.
- $p(\mathcal{I}|\mathcal{F}, \mathcal{B}, \tau)$ est le terme de vraisemblance des images. Les dépendances considérées reflètent le fait que l'ensemble des observations dans les images n'est conditionné que par la détection de silhouette dans les images, et par la connaissance des images du fond.

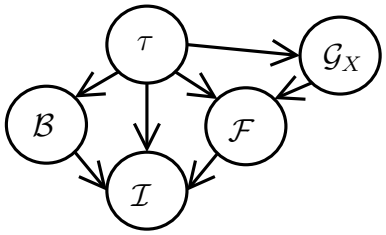


FIG. 4.1 – Les variables du système proposé, et leur graphe de dépendances. τ : Symbolique des éléments *a priori* introduits dans le modèle. $\mathcal{G}_X$: occupation au voxel X. $\mathcal{B}$: carte des paramètres du modèle du fond. $\mathcal{F}$: carte de détection des silhouettes. $\mathcal{I}$: images observées.

4.3.1 Simplifications de la fusion de capteurs

Les couleurs des pixels des images d'entrée sont traitées comme des observations bruitées du modèle. Nous supposons, comme généralement dans un problème de fusion de capteurs, que le bruit est indépendamment et identiquement distribué. De plus, nous supposons que les observations de couleur obtenues en chaque pixel ne sont expliquées que par les images du fond ainsi que l'état de la détection de silhouette *en ce même pixel*. Ceci induit une indépendance conditionnelle entre

toutes les observations de couleur $\mathcal{I}_p^i$:

$$p(\mathcal{I}|\mathcal{F},\mathcal{B},\tau) = \prod_{i,p} p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)$$

Nous exprimons que chaque variable de détection de silhouette ne dépend elle-même que de la connaissance que nous avons de l'état d'occupation de la grille, et qu'elle est indépendante de toute autre variable capteur. C'est également une hypothèse courante des problèmes de fusion de capteur : toutes les détections de silhouette sont conditionnellement indépendantes, sachant leur principale cause, à savoir l'occupation des voxels :

$$p(\mathcal{F}|\mathcal{G}_X,\tau) = \prod_{i,p} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau)$$

En conséquence, la distribution conjointe des variables d'intérêt est réduite au produit suivant :

$$p(\mathcal{G}_X,\mathcal{I},\mathcal{B},\mathcal{F},\tau) = \prod_{i,p} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau)p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau) \tag{4.1}$$

Nous avons donc réduit l'évaluation de la probabilité conjointe de toutes les images, de toutes les cartes de détection, et de notre occupation au voxel considéré, à deux sous-problèmes beaucoup plus simples. D'une part, l'expression de la vraisemblance de la détection de silhouette en un seul pixel, sachant l'occupation de notre voxel. C'est le terme de formation des silhouettes (paragraphe 4.3.2). D'autre part, l'expression de la vraisemblance d'une observation couleur, sachant l'état de détection en ce pixel, et les paramètres du fond en ce pixel. Il s'agit du terme de formation des images (paragraphe 4.3.3). Penchons-nous maintenant sur ces deux termes.

4.3.2 Terme de formation des silhouettes

Le terme de formation des silhouettes $p(\mathcal{F}_p^i|\mathcal{G}_X,\tau)$ modélise la réponse de la détection d'un pixel (i,p) à l'état d'occupation de notre voxel d'intérêt $\mathcal{G}_X$. Nous avons besoin d'introduire deux variables cachées locales $\mathcal{S}$ et $\mathcal{R}$ pour tempérer l'influence de ce voxel. La figure 4.2 présente les variables et dépendances statistiques de ce sous-problème. Dans un cas idéal et dénué de bruit, les deux variables $\mathcal{F}_p^i$ et $\mathcal{G}_X$ seraient auto-suffisantes, et liées par une relation de logique booléenne : si notre voxel X est occupé, et s'il se projette au point p, alors il y a une détection de silhouette au pixel p, $\mathcal{F}_p^i = 1$. C'est la formulation implicite utilisée dans les méthodes classiques d'enveloppe visuelle.

Cependant, il existe des sources d'incertitude qui viennent perturber ce raisonnement intuitif. D'une part, le fait qu'un voxel se trouve sur la ligne de vue d'un

pixel est lui-même incertain. Ceci peut-être dû à de nombreuses causes externes : les petites erreurs de calibrage, la disparité des instants d'acquisition des images, qui introduisent des défauts d'alignement dans la scène. Il y a aussi des problèmes d'échantillonnage, car aucun voxel ne se projette parfaitement sur un pixel : sa surface de projection peut en couvrir plusieurs. D'autre part, des phénomènes autres que l'occupation du voxel lui-même peuvent expliquer la détection : une occupation due à un autre voxel que X, ou un changement d'apparence du fond (une cause *interne* de défaillance au vu du modèle capteur que nous définissons).

Il est possible de modéliser ces phénomènes en utilisant deux variables cachées booléennes $\mathcal{S}$ et $\mathcal{R}$. Ceci nous conduit à deux expressions du terme de détection de silhouette $p(\mathcal{F}_p^i|\mathcal{G}_X, \tau)$. Tout d'abord, considérons le cas où nous savons que notre voxel X est occupé ($\mathcal{G}_X = 1$) :

$$\begin{aligned} p(\mathcal{F}_p^i|[\mathcal{G}_X{=}1], \tau) &= p(\mathcal{S}{=}0|\tau)\ \mathcal{U}(\mathcal{F}_p^i) \\ &+ p(\mathcal{S}{=}1|\tau)\ \mathcal{P}_d(\mathcal{F}_p^i) \end{aligned} \tag{4.2}$$

Par définition, la *variable d'échantillonnage* $\mathcal{S}$ est égale à 1 si le voxel X se trouve sur la ligne de vue du pixel (i, p). Si ce n'est pas le cas ($\mathcal{S} = 0$), alors la connaissance de l'occupation de notre voxel ne nous apporte aucune information sur la réponse du pixel. Ceci explique la distribution uniforme $\mathcal{U}(\mathcal{F}_p^i)$ pour la détection de silhouette dans l'expression (4.2). Si le voxel se trouve sur la ligne de vue de p ($\mathcal{S} = 1$), alors la détection à ce pixel est régie par la loi de probabilité $\mathcal{P}_d(\mathcal{F}_p^i)$. En pratique nous définissons cette distribution en utilisant une constante $P_D \in [0, 1]$, qui est un paramètre de notre système : $\mathcal{P}_d([\mathcal{F}_p^i = 1]) = P_D$ est le taux de détection d'un pixel en tant que capteur, et réciproquement $\mathcal{P}_d([\mathcal{F}_p^i = 0]) = 1 - P_D$ est son taux de défaut de détection. Une telle défaillance a lieu lorsque le pixel relate de manière erronée l'absence de matière sur sa ligne de vue. Ceci est fondamentalement utile pour notre problème : en effet il peut arriver que la détection de silhouette échoue localement dans une image. Le fait de modéliser explicitement cet état de fait au niveau de notre capteur donne la possibilité à notre système de corriger une telle erreur grâce à la contribution d'autres images.

Considérons maintenant le cas où l'on sait que notre voxel n'est pas occupé ($\mathcal{G}_X = 0$) :

$$\begin{aligned} p(\mathcal{F}_p^i|[\mathcal{G}_X{=}0], \tau) = p(\mathcal{S}{=}0|\tau)\ &\mathcal{U}(\mathcal{F}_p^i) \\ + p(\mathcal{S}{=}1|\tau)\ \big[\ &p(\mathcal{R}{=}1|\tau)\ \mathcal{P}_d(\mathcal{F}_p^i) \\ &+ p(\mathcal{R}{=}0|\tau)\ \mathcal{P}_f(\mathcal{F}_p^i)\ \big] \end{aligned} \tag{4.3}$$

Comme dans le cas précédent, lorsque le voxel ne se trouve pas sur la ligne de vue de p (cas où $\mathcal{S} = 0$), on ne peut rien conclure sur la détection en ce pixel.

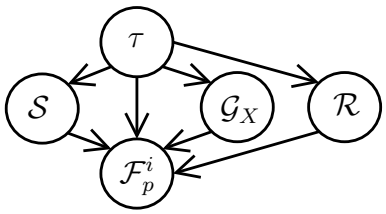


FIG. 4.2 – Variables et graphe de dépendances du sous-problème de détection de silhouette en un pixel. τ : connaissance *a priori*. $\mathcal{G}_X$: occupation de notre voxel. $\mathcal{S}$: variable d'échantillonnage. $\mathcal{R}$: variable modélisant les causes externes de détection. $\mathcal{F}_p^i$: détection de silhouette au pixel (i, p).

Cependant même si le voxel se trouve sur la ligne de vue de ce pixel (cas où $\mathcal{S} = 1$), il n'est toujours pas possible de conclure sur l'état de la détection. Soit une variable de "causes extérieures" $\mathcal{R}$, valant 1 si un autre objet se trouve sur la même ligne de vue que notre voxel. Dans un tel cas, la détection est de nouveau régie par la loi de probabilité $\mathcal{P}_d(\mathcal{F}_p^i)$. Cependant, dans le cas où aucun autre objet n'obstrue cette ligne de vue ($\mathcal{R} = 0$), la loi de probabilité de détection est régie par la distribution $\mathcal{P}_f(\mathcal{F}_p^i)$. Nous établissons cette distribution à l'aide d'une constante $P_{FA} \in [0, 1]$, qui est un paramètre de notre système : $\mathcal{P}_f([\mathcal{F}_p^i = 1]) = P_{FA}$ est le taux de fausse alarme d'un pixel. C'est le taux avec lequel le pixel relate faussement la présence de matière le long de sa ligne de vue, lorsqu'elle est en fait absente. $\mathcal{P}_f([\mathcal{F}_p^i = 0]) = 1 - P_{FA}$ est le taux de non-détections correctes et avérées.

Nous devons donner une forme paramétrique à $p(\mathcal{R}|\tau)$. Il peut y avoir des causes de détection partout sur la ligne de vue de p. Nous ne faisons aucune supposition sur la nature de ces causes et considérons que la détection peut être provoquée aussi bien par l'occupation d'autres voxels de la grille ou par ces causes. Ceci nous conduit à affecter une distribution uniforme à ce terme. Ce faisant, nous considérons que le plus important n'est pas forcément de donner une modélisation très élaborée à ces termes, mais que c'est le fait de laisser à l'inférence la possibilité de prendre en compte la présence de ces phénomènes extérieurs qui est déterminant en soi.

Forme paramétrique pour le terme d'échantillonnage $p(\mathcal{S}|\tau)$. Ce terme dépend de i, p et X. Nous utilisons un échantillonnage uniforme, avec $p(\mathcal{S}|\tau) = \mathcal{U}_{k\times k}(x - p)$, x étant le point image projeté de X dans l'image i en question. Ceci donne un poids équivalent à tout les voxels se trouvant dans une fenêtre de taille $k \times k$ autour du pixel p. Une fonction d'échantillonnage plus lisse (de forme Gaussienne) aurait aussi pu être utilisée, mais mobilise des ressources de calcul plus importantes pour intégrer l'information. De manière générale, la forme de cette fonction peut aisément être adaptée pour répondre à des besoins spécifiques.

La fonction d'échantillonnage permet d'avoir un contrôle sur les petites erreurs

de calibrage, de synchronisation entre les caméras, et sur les erreurs de classification dans les images : en effet elle permet à plusieurs pixels de participer à la classification du même voxel au cours de l'inférence. Grâce à l'introduction de ces deux variables cachées et de leur forme paramétriques, notre méthode unifie la gestion de l'incertitude dans le cadre des silhouettes et les schémas d'échantillonnage classiques utilisés dans certaines méthodes à base d'enveloppe visuelle [22].

4.3.3 Terme de formation des images

Le but du terme de formation des images $p(\mathcal{I}_p^i|\mathcal{F}_p^i, \mathcal{B}_p^i, \tau)$ est de fournir un modèle qui explique l'information de couleur au pixel (i, p), connaissant les paramètres du modèle statistique du fond à ce pixel, et selon l'état de la détection à ce pixel. Nous exposons ici les formes paramétriques que nous choisissons de donner à ces termes.

Considérons tout d'abord le cas où une détection de silhouette a eu lieu au pixel (i, p). La connaissance du fond ne nous apporte aucune information supplémentaire sur la couleur que l'on peut espérer observer à ce pixel. Nous savons en effet que les objets du fond sont occultés par un objet d'intérêt, dont le pixel observe la couleur. Nous ne faisons aucune supposition sur la couleur des objets d'intérêt, ce qui nous conduit à donner une loi uniforme à la distribution des couleurs observées ici :

$$p(\mathcal{I}_p^i|[\mathcal{F}_p^i = 1], \mathcal{B}_p^i, \tau) = \mathcal{U}(\mathcal{I}_p^i)$$

Le second cas que nous devons considérer est celui où aucune silhouette n'a été détectée en ce pixel. Intuitivement, si l'on sait que le capteur ne voit pas de matière sur sa ligne de vue, alors la couleur observée en ce pixel doit ressembler à celle du fond. En pratique, nous choisissons de résumer toutes les images du fond dont nous disposons en estimant les paramètres $\mathcal{B}_p^i = (\mu_p^i, \sigma_p^i)$ d'une distribution normale dans l'espace de couleur (Y,U,V), pour chaque pixel. Nous pouvons alors formuler pour le pixel (i, p) le raisonnement exprimé ci-dessus de la manière suivante, à l'aide de cette distribution :

$$p(\mathcal{I}_p^i|[\mathcal{F}_p^i = 0], [\mathcal{B}_p^i = (\mu_p^i, \sigma_p^i)], \tau) = \mathcal{N}(\mathcal{I}_p^i|\mu_p^i, \sigma_p^i)$$

Cette représentation des couleurs du fond dérive des méthodes classiques de soustraction de fond [104]. Il est important de souligner cependant que l'outil présenté ici est indépendant du modèle statistique choisi pour représenter le fond. Il pourrait facilement utiliser d'autres représentations du fond, comme le mélange de Gaussiennes [88, 43, 96], qui est plus robuste aux ambiguïtés sous-pixelliques et aux variations périodiques de couleur dans les images du fond. Néanmoins, certains problèmes persistent quel que soit le modèle de couleur utilisé : les ambiguïtés de couleur entre les objets d'intérêt et ceux du fond, les changement dans la géométrie

ou l'illumination de la scène. C'est le but de notre approche multi-caméra de compenser et d'être plus robuste à de tels problèmes, qui sont difficiles à corriger dans un contexte monoculaire.

4.4 Inférence sur l'occupation d'un voxel

Maintenant que la distribution conjointe est complètement déterminée, il est possible d'utiliser la règle de Bayes pour inférer la distribution de probabilité de notre variable *recherchée* $\mathcal{G}_X$, sachant la valeur de nos variables *connues* $\mathcal{I}, \mathcal{B}, \tau$, et en marginalisant l'inférence par rapport à nos variables *inconnues* $\mathcal{F}$:

$$\begin{aligned} p(\mathcal{G}_X|\mathcal{I},\mathcal{B},\tau) &= \frac{\sum_{\mathcal{F}} p(\mathcal{G}_X,\mathcal{I},\mathcal{B},\mathcal{F},\tau)}{\sum_{\mathcal{G}_X,\mathcal{F}} p(\mathcal{G}_X,\mathcal{I},\mathcal{B},\mathcal{F},\tau)} \\ &= \frac{\sum_{\mathcal{F}} \prod_{i,p} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau) p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)}{\sum_{\mathcal{G}_X,\mathcal{F}} \prod_{i,p} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau) p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)} \qquad (4.4) \\ &= \frac{\prod_{i,p} \sum_{\mathcal{F}_p^i} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau) p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)}{\sum_{\mathcal{G}_X} \prod_{i,p} \sum_{\mathcal{F}_p^i} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau) p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)} \qquad (4.5) \end{aligned}$$

où (4.1) a été substituée dans (4.4). Cette inférence est simplifiée du fait que toutes les sommes sur les variables $\mathcal{F}_p^i$ peuvent être réalisées au niveau de chaque pixel et donc factorisées (4.5). En particulier, $\mathcal{F}_p^i$ peut elle-même être considérée comme une variable cachée intervenant au niveau du pixel. On peut dans ce cadre considérer que le modèle capteur se résume à un seul terme générique de formation des couleurs dans les images $p(\mathcal{I}_p^i|\mathcal{G}_X,\mathcal{B}_p^i,\tau)$. Ce terme lie directement la couleur observée en chaque pixel à l'état d'occupation de notre voxel. Il peut être exprimé à l'aide des termes de formation des silhouettes et des images présentés ci-dessus. Il suffit alors de marginaliser la détection de silhouette $\mathcal{F}_p^i$, comme écrit ci-dessous :

$$p(\mathcal{I}_p^i|\mathcal{G}_X,\mathcal{B}_p^i,\tau) = \sum_{\mathcal{F}_p^i} p(\mathcal{F}_p^i|\mathcal{G}_X,\tau) p(\mathcal{I}_p^i|\mathcal{F}_p^i,\mathcal{B}_p^i,\tau)$$

Cette vue du modèle capteur clarifie l'inférence (4.5), car elle montre comment chaque pixel contribue à la distribution de probabilité d'occupation de notre voxel :

$$p(\mathcal{G}_X|\mathcal{I},\mathcal{B},\tau) = \frac{\prod_{i,p} p(\mathcal{I}_p^i|\mathcal{G}_X,\mathcal{B}_p^i,\tau)}{\sum_{\mathcal{G}_X} \prod_{i,p} p(\mathcal{I}_p^i|\mathcal{G}_X,\mathcal{B}_p^i,\tau)} \qquad (4.6)$$

Notons que l'expression finale de l'inférence (4.6) lie, dans son écriture, l'occupation d'un voxel à *tous* les pixels et leur observations. Un tel calcul est bien sûr inenvisageable, sachant qu'une telle inférence doit être réalisée *pour chaque voxel* de la grille. En pratique, les termes d'échantillonnage présentés au paragraphe

4.3.2 permettent de borner la région d'influence d'un pixel dans les images. Les expressions des probabilités de détection des pixels trop éloignés de la projection d'un voxel dégénèrent en un terme uniforme, ce que les équations (4.2) et (4.3) expriment. Dans ce cas la contribution peut être factorisée et éliminée de l'expression d'inférence (4.6). Ceci rend l'inférence praticable, en calculant ce produit sur une fenêtre locale de pixels, centrée autour du point de projection de X, dans chaque image. Notons que le calcul d'un tel produit est rapidement limité par la capacité de représentation des nombres de la machine : en pratique nous calculons ce produit en utilisant une somme de log probabilités. Si k est la taille de la fenêtre, et N le nombre de voxels par dimension de l'espace, alors la complexité en calcul de l'inférence de tous les voxels de la grille est $O(n\ k^2 N^3)$. L'utilisation d'un échantillonnage uniforme permet une optimisation (non implémentée ici), en précalculant des produits de termes sur toute l'image pour calculer astucieusement le produit sur une fenêtre de pixels en un nombre borné d'accès mémoire. Avec une telle optimisation, la complexité du calcul peut être ramenée à $O(n\ N^3)$.

4.5 Résultats et applications

Nous avons implémenté cet algorithme de fusion d'information silhouette multi-vue, en utilisant une stratégie d'échantillonnage uniforme pour les voxels dans nos expérimentations. Comparé à un échantillonnage Gaussien ceci s'est avéré être un bon compromis entre coût de calcul et capacité d'intégration de l'information de plusieurs pixels. Notons que la méthode dans son ensemble ne possède que trois paramètres $\{P_D, P_{FA}, k\}$, le taux de détection, le taux de fausse alarme, et la taille de la fenêtre d'échantillonnage. Souvent ces paramètres peuvent être fixés une fois pour toute pour une application donnée. En pratique, P_D et P_{FA} pondèrent la confiance accordée aux observations. Si $P_{FA} = 0$ et $P_D = 1$, nous accordons en ces observations une confiance aveugle. Si P_{FA} et P_D sont proches de 0.5 alors nos observations ne sont pas fiables : on a besoin de beaucoup plus d'observations pour conclure sur l'état d'occupation d'un voxel. k permet de décider du nombre d'observations à considérer localement dans chaque image et permet donc aussi un contrôle de la décision et de la robustesse de la méthode aux petites erreurs. L'algorithme peut être utilisé dans de nombreux champs d'application. Nous l'avons testé dans des conditions très diverses.

4.5.1 Modélisation à partir d'images

La grille calculée fournit une estimation de forme. Nous illustrons ce fait dans la séquence dite de marche. Il s'agit de l'acquisition d'une scène où une personne est en train de marcher en cercle, réalisée avec 8 caméras de caractéristiques

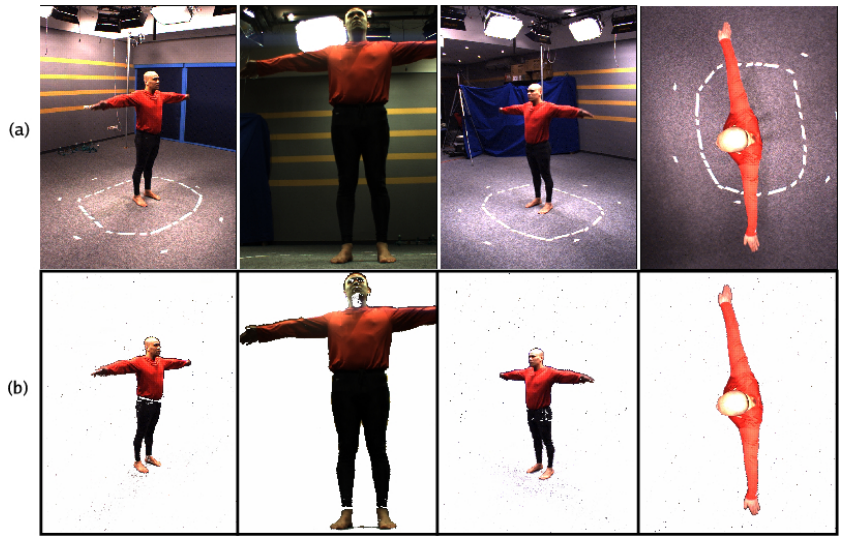


FIG. 4.3 – Entrées de l'algorithme. (a) Quatre des huit images de la séquence de marche (8 caméras, 15 images/sec) (b). Le résultat d'une soustraction de fond monoculaire en utilisant le même modèle de représentation des couleurs que dans notre algorithme (rendu semi-transparent pondéré par la probabilité des silhouettes). Notons les imperfections de ces silhouettes : la deuxième caméra ne voit pas l'avant bras gauche de la personne. Il y a des trous et du bruit dans les silhouettes de plusieurs vues.

et résolutions différentes (640x480, 780x580) avec une fréquence d'acquisition de 15Hz. Comme l'illustre la figure 4.3 l'information silhouette accessible par soustraction de fond monoculaire est bruitée et comporte des erreurs. Notons aussi que certaines caméras ne voient pas toujours entièrement la personne au cours de la séquence. Ceci est le cas dans la deuxième vue de la figure, où l'avant bras de la personne est coupé. Le modèle de couleur utilisé pour les soustractions de fond monoculaires est le même que celui utilisé dans notre modèle (distribution normale). Ces soustractions de fond résument l'information silhouette dont notre algorithme dispose dans chaque image.

La figure 4.4 montre le résultat de notre méthode à un instant de la séquence de marche, en utilisant une grille de 120^3 voxels. La figure comporte des coupes verticales et horizontales de la grille, qui montrent la nature de l'information disponible dans celle-ci.

Comme le montre la figure 4.4(c), la méthode fournit de bons résultats de modélisation, si l'on extrait une isosurface de la grille de probabilités. La méthode donne une surface finement détaillée, avec de l'information sous-voxélique. Les petits trous qui apparaissaient dans les soustractions de fond monoculaires sont comblés. D'autres résultats de modélisation apparaissent dans la figure 4.5. Dans

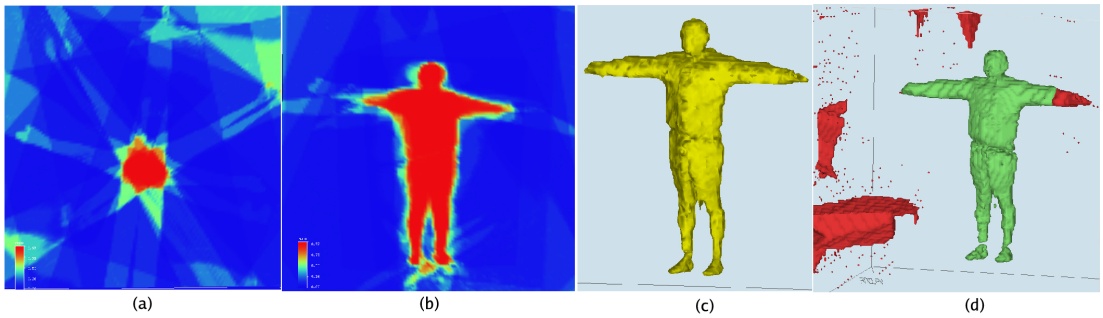


FIG. 4.4 – Un instant de la séquence de marche. Acquisition 15imgs/sec, 8 caméras, utilisant en entrée les données explicitées en figure 4.3. Grille de 120^3 voxels. Temps de calcul : approximativement 10 sec sur un PC 2.4 GHz. Paramètres utilisés : $P_D = 0.9, P_{FA} = 0.1, k = 5$ (a) Coupe horizontale (au niveau de l'abdomen) dans la grille de probabilité. Les régions vertes au coin supérieur droit ne sont vues par aucune caméra (probabilité 0.5). (b) Coupe verticale de la grille. (c) Isosurface de probabilité 0.80, obtenue de la grille. Le bruit dû aux ombres ne perturbe pas l'estimation, la majorité des défauts des silhouettes sont comblés. (d) Deux approches classiques de reconstruction à base d'enveloppe visuelle : en vert, sous l'hypothèse que toutes les caméras voient entièrement l'objet. L'avant-bras gauche est perdu. En rouge, sous l'hypothèse que les voxels en dehors du domaine de visibilité d'une caméra peuvent faire partie de l'enveloppe visuelle. Le bras n'est cette fois-ci pas coupé, mais des objets fantômes apparaissent, résultant d'ambiguïtés visuelles à des endroits où toutes les caméras ne voient pas les voxels.

nos résultats, le seuil est choisi à la main. Il s'agit d'une amélioration appréciable par rapport au schéma de reconstruction géométrique qui nécessite de fixer autant de seuils qu'il y a d'images pour identifier les silhouettes. Bien que cette possibilité n'ait pas été explorée dans cette thèse, il serait néanmoins possible, à partir des informations de la grille de probabilités, de choisir un seuil de manière automatique. Une méthode d'optimisation globale telle une coupure optimale de graphe peut être utilisée à ces fins.

L'approche voxélique classique avec silhouettes binaires a été implémentée pour comparaison, et les résultats sont montrés en figure 4.4(d). Chaque voxel y est projeté dans les images et sculpté si cette projection est en dehors des silhouettes. Nous utilisons la soustraction de fond de la figure 4.3 pour cette expérience, et sélectionnons manuellement le meilleur seuil *dans chaque image* pour fournir les silhouettes binaires nécessaires à cet algorithme. Les trous qui apparaissent dans les silhouettes engendrent en général des trous et défauts dans le volume reconstruit. Notons que notre méthode retrouve une information d'occupation valide avec des vues qui ne voient pas entièrement l'objet. Ceci est transparent pour notre algorithme, car celui-ci n'intègre que l'information des capteurs qui voient

les voxels.

Ceci est différent de toute approche classique à base d'enveloppe visuelle, aussi bien surfacique que volumétrique. En effet, celles-ci font toutes des hypothèses explicites qui forcent la décision pour les voxels se trouvant en dehors de la zone de visibilité d'une caméra. La figure 4.4(d) montre le résultat du calcul purement géométrique de l'enveloppe visuelle, avec en vert le résultat obtenu en supposant que tous les objets sont vus de toutes les caméras (selon la définition de l'expression 3.1 du paragraphe 3.2.4), et en rouge le résultat obtenu en construisant l'enveloppe visuelle en utilisant la définition de l'expression 3.1 du paragraphe 3.2.4, qui fait apparaître des objets fantômes. Les objets fantômes sont bien sous-jacents à la représentation probabiliste proposée (comme on peut le voir dans la figure 4.4(a), dans les régions périphériques de la grille), mais restent la plupart du temps moins probables que l'objet réel car la probabilité d'une région est d'autant plus forte et décidée qu'elle est observée par un nombre élevé de caméras. Les objets fantômes étant par définition dans des zones en dehors du champ de certaines caméras, ils sont donc nécessairement moins probables que l'objet lui-même, observé ici par quasiment toutes les vues. Il est donc généralement possible, grâce à la méthode probabiliste proposée, de séparer l'objet réel des objets fantômes en sélectionnant un seuil approprié.

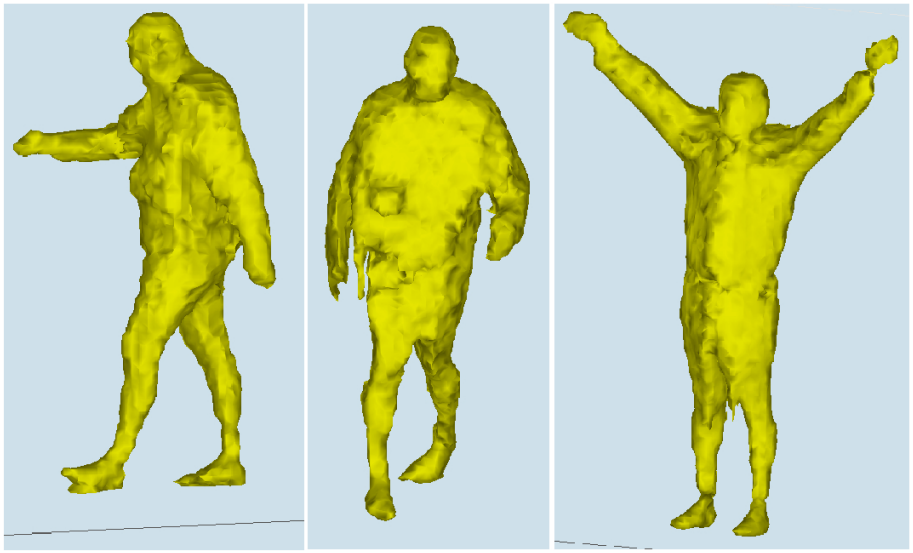


FIG. 4.5 – Isosurface de probabilité 0.80 à différents instants de la séquence de marche.

4.5.2 Soustraction de fond multi-vue

Notre méthode permet d'effectuer une fusion d'informations de silhouette. Cette fusion peut être utilisée pour calculer des silhouettes cohérentes dans les images. En effet, il est possible de reprojeter et effectuer un rendu de la grille d'occupation, en projetant le maximum d'intensité de la grille (voir figure 4.6). Cette heuristique est une approximation du calcul d'inférence de l'état de détection de la silhouette d'un pixel sachant toutes les observations images, calcul possible mais pratiquement infaisable au vu de sa complexité. La projection du maximum d'intensité nous permet de localiser, à un coût moindre, les zones où les silhouettes sont plus probables dans les images.

Cette heuristique de rendu défini une procédure de soustraction de fond multi-caméra. En particulier, un seul seuil peut-être choisi simultanément pour toutes les images pour trouver les silhouettes binaires associées, comme le montre la figure. Les détails fins de la silhouette sont préservés, et sont seulement limités par la résolution de la grille. Chaque vue bénéficie alors de la connaissance des informations silhouette des autres vues.

Dans la figure 4.6, nous pouvons voir deux exemples de rendu de silhouettes multi-vue. En figure 4.6(a) sont montrés les résultats de la soustraction de fond monoculaire (rendu semi-transparent pondéré par la probabilité). On peut remarquer les imperfections : coupure de la silhouette au niveau du bassin dans l'image du dessus ; pieds séparés du corps, et erreur grossière au niveau du visage, dans l'image du dessous. La correction proposée est plus fine qu'une simple correction mono-image par opération morphologique : elle bénéficie simultanément de toute l'information des images. La figure 4.6(b) montre la projection du maximum d'intensité de la grille d'occupation (120^3) depuis les points de vues de l'acquisition. La figure 4.6(c) montre les même données après sélection manuelle d'un seuil *pour toutes les silhouettes simultanément* : les silhouettes sont améliorées par rapport aux silhouettes monoculaires initiales. Des artefacts d'échantillonnage apparaissent selon la résolution de la grille et la configuration de la scène.

4.5.3 Détection d'objet

La méthode peut être utilisée dans des conditions beaucoup plus dures pour inférer de l'information sur une scène. En particulier, en présence de hauts niveaux de bruit, la taille de la fenêtre d'échantillonnage peut être augmentée pour une robustesse accrue, avec cependant un impact négatif sur la précision (cette opération tend en effet à dilater le volume dans la grille, et par conséquent les isosurfaces que l'on peut en extraire). Lorsqu'il y a beaucoup de bruit dans les images, l'utilisation de la méthode peut devenir limitée pour les applications de modélisation

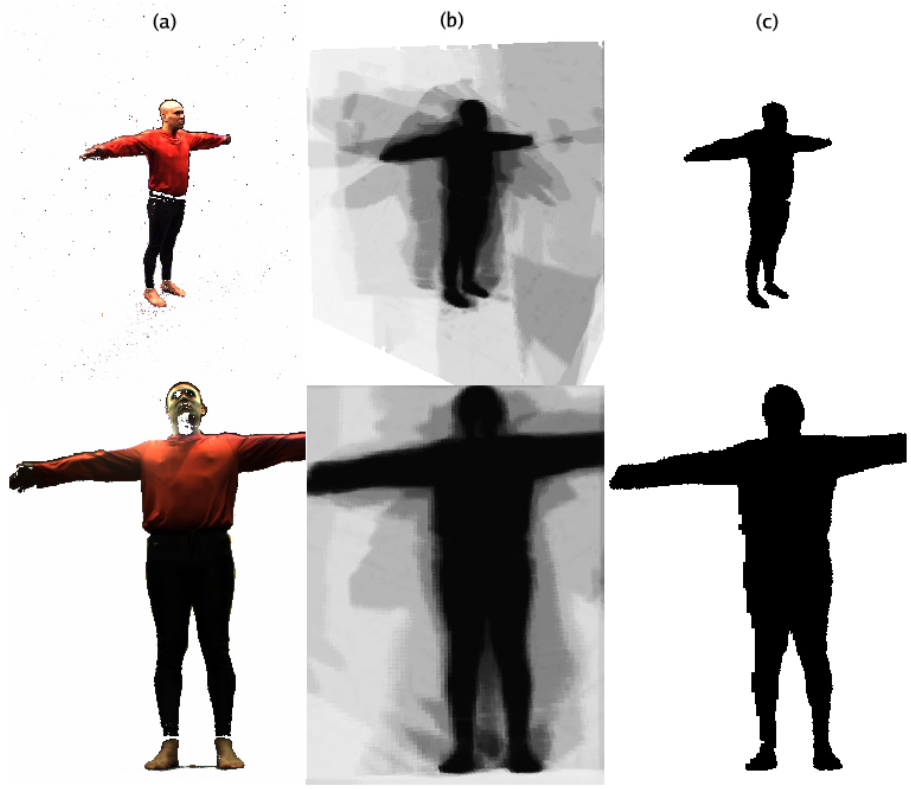


FIG. 4.6 – Deux exemples de rendu de silhouettes multi-vue.

3D. Cependant la méthode peut encore être utilisée pour *localiser* les objets dans une scène, sans nécessairement avoir pour but de reconstruire précisément leur surface. Nous illustrons l'utilisation potentielle de notre méthode dans le cadre d'une application de localisation, avec des caméras non précisément réglées, et des images avec un contraste pauvre, dans la figure 4.7. Dans cette expérience, 8 caméras ont été placées dans une scène de manière à couvrir et surveiller une zone relativement étendue ($25m^2$) dans une pièce. Seul le centre de la pièce est vu par une majorité de caméras. Les zones périphériques de l'espace d'acquisition ne sont vues que par trois ou quatre caméras tout au plus.

Deux personnes marchent aléatoirement dans la pièce et sont localisées avec succès, lorsqu'elles sont vues par au moins trois caméras. Un suivi d'objet très simple peut être implémenté sur la base de cette méthode en identifiant les composantes connexes de voxels dont la probabilité est supérieur à un seuil fixé, par exemple, mais peut aussi fournir la base pour des méthodes de suivi plus élaborés. Le résultat empirique est intéressant : deux caméras ne semblent pas être suffisantes pour apporter l'information nécessaire à la décision, et distinguer les vrais objets d'objets "fantômes" qui apparaissent dans les zones d'ambiguïté visuelle. Ces régions d'ambiguïté sont des régions vides de l'espace mal détectées par les

caméras, qui sont en général dans l'ombre d'un véritable objet, par rapport au point de vue d'une ou plusieurs caméras.

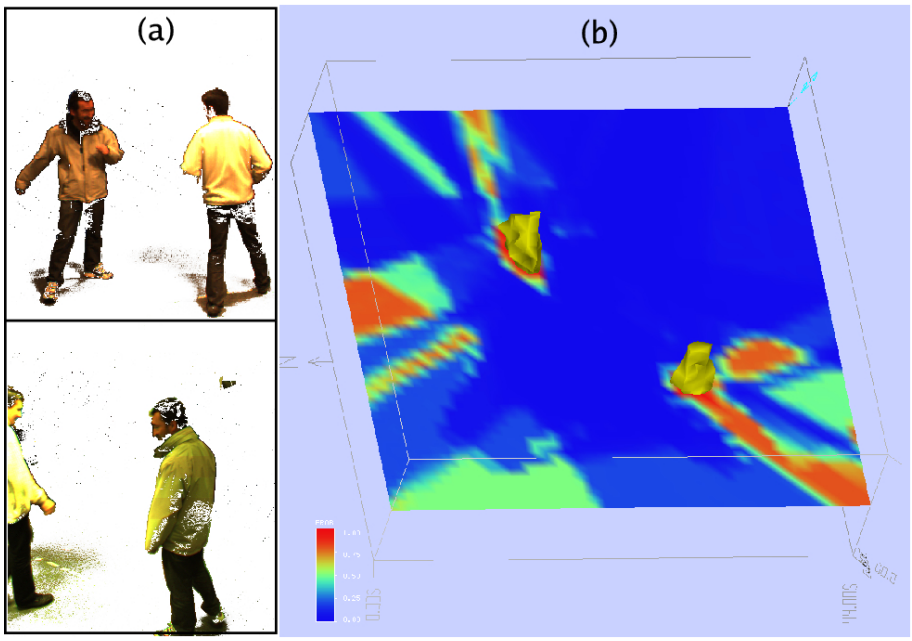


FIG. 4.7 – Séquence multi-objet, avec 8 caméras. (a) Soustractions de fond monoculaires de quelques vues d'entrée (rendu semi-transparent pondéré par la probabilité d'appartenir à une silhouette). Des conditions difficiles de lumières et contraste créent de grandes difficultés pour de telles soustractions monoculaires. (b) Notre méthode, utilisée pour reconstruire une grille grossière de la scène (50x50x18), suffisante pour localiser des objets (avec $k = 25$). Temps de calcul pour un instant de la séquence : 7s. Une coupe horizontale de la grille de probabilités est donnée, ainsi que des isosurfaces de probabilité 0.67 qui montrent les objets localisés. (c) Configuration des caméras dans cette scène. Les régions en rouge sombre sont moins fiables car elles ne sont vues que par un maximum de 2 caméras. Le système est capable de détecter la présence d'objets dans une zone de 5m x 5m.

4.6 Prise en compte de la dynamique

Nous avons présenté une méthode permettant d'inférer une structure probabiliste de la scène, en fonctions d'observations de couleurs de pixels dans les images à un instant donné, et de modèles de fond préalablement observés pour chaque vue. Comme nous l'avons constaté dans le précédent paragraphe, la résolution du problème statique à chaque instant fonctionne et fournit déjà des résultats parfaitement exploitables pour l'estimation de forme à partir de silhouettes incertaines. Cependant, il apparaît intuitivement que l'usage d'observations à différents instants nous apporterait plus d'information pour la résolution de notre problème, qui est dynamique par nature. Plusieurs méthodes ont déjà été proposées pour tirer parti de cette information supplémentaire. Vedula *et al.* [103] proposent une méthode pour sculpter des voxels avec une contrainte photométrique plus forte qu'un simple Space Carving, liant deux reconstructions consécutives de la séquence. Le résultat formalise, de manière relativement limitée, l'idée que le volume se déplace et se déforme de manière contrainte entre des instants consécutifs de l'acquisition. Cette notion sera reprise et reformulée dans l'étude que nous proposons. Cheung, Baker & Kanade [21] proposent une méthode de raffinement temporel de l'enveloppe visuelle en utilisant une méthode d'alignement photométrique. La méthode permet de reconstruire l'enveloppe visuelle d'un objet rigide soumis à un mouvement rigide, observé avec un nombre de points de vues accumulé au cours du temps, mais nécessite une initialisation très lourde pour identifier la structure du modèle articulé de la personne. Goldlücke & Magnor [46] proposent une méthode variationnelle pour sculpter photométriquement la matière tout en lissant la surface reconstruite sur l'ensemble de la séquence. La méthode donne des résultats intéressants mais est très coûteuse en temps de calcul et se limite par définition au traitement hors-ligne. L'ensemble des méthodes abordées se sert de l'information photométrique pour contraindre et identifier la cohérence temporelle. Nous montrons ici la possibilité d'une solution dynamique de filtrage, peu gourmande en ressources, se servant uniquement d'informations liées aux silhouettes observées.

L'approche par grille d'occupation nous a paru doublement appropriée pour ouvrir de nouvelles possibilités : premièrement parce qu'elle est associée à un formalisme probabiliste du problème, qui nécessite de se pencher sur l'ensemble du processus d'estimation et de mesure, et sur le fonctionnement du pixel en tant que capteur en particulier. Deuxièmement parce qu'elle admet une extension naturelle pour le cas dynamique, qui est fortement exploitée en robotique [27]. L'extension se présente comme un processus de filtrage temporel, réalisé via une boucle de prédiction-estimation sur l'état des voxels, et implique donc de traiter un problème difficile : celui de modéliser la propagation d'un volume échantillonné sur une grille, d'un pas de temps au suivant. Nous nous penchons sur cette extension

dans le cadre de notre problème de vision, et fournissons un résultat préliminaire permettant de se rendre compte des possibilités offertes par la grille d'occupation, mais aussi des limitations liées à la particularité de notre problème.

4.6.1 Reformulation du problème

Nous considérons toujours le problème d'estimation de forme à partir d'observations défini au paragraphe 4.2. L'extension du problème au cas dynamique implique de se pencher non plus sur un jeu de variables à un instant donné, mais sur $T+1$ jeux de variables correspondant aux grandeurs considérées à différents instants d'un horizon de temps discret $t \in \{0, \cdots, T\}$. Dans les prochains paragraphes, nous noterons en exposant l'indice temporel t caractérisant les variables à l'instant t. Nous utiliserons spartiatement le numéro d'image, lui aussi noté en exposant, pour faciliter la lecture. Nous ne ferons intervenir celui-ci que dans les expressions finales où celles-ci auront un sens, et préciserons à ce moment-là les notations.

Variables du problème

Nous ne nous intéressons plus à un seul état d'occupation de notre grille de points mais à une famille d'états de la grille $\mathcal{G}^t, t \in \{0, \cdots, T\}$, notée par souci de clarté $\mathcal{G}^{0:T}$. Nous étendons cette notation pour toutes les autres variables du problème. En particulier, nous notons les familles d'images observées $\mathcal{I}^{0:T}$. A chaque instant t correspond donc un jeu de N images noté $\mathcal{I}^t$, correspondant aux N vues de la scène. Nous avons préalablement construit un modèle $\mathcal{B}$ du fond pour chacune des vues, alors dénuées d'objet d'intérêt. Nous faisons l'hypothèse d'un fond statique, et donc que $\mathcal{B}$ ne dépend pas du temps. τ résume toujours nos connaissances *a priori* sur le système. Comme τ et $\mathcal{B}$ apparaissent toujours en tant que variables connues, et sont supposées a priori uniformes, nous les omettons de toute équation suivante pour alléger l'écriture. Il s'agit donc d'estimer à chaque instant t de la séquence, l'état de la grille $\mathcal{G}^t$ le plus probable.

Distribution conjointe

Pour simplifier l'explication, nous nous plaçons au niveau le plus global du problème, en ne considérant pour l'instant que les variables recherchées et les variables connues du problème, c'est à dire les familles de variables d'occupation et d'observation ci-dessus. Nous avons vu en particulier dans le traitement du cas statique que toutes les autres variables du problème sont des variables cachées, donnant lieu à une sommation pouvant être regroupée dans un terme *par pixel* (y compris pour les cartes de silhouettes, vu en paragraphe 4.4). Ceci permet de traiter ces variables dans le cadre d'un sous-problème Bayésien, comme nous

l'avons fait au paragraphe 4.3.2 pour nuancer le terme de détection des silhouettes. Nous nous intéressons donc à la distribution conjointe de variables $p(\mathcal{G}^{0:T}\ \mathcal{I}^{0:T})$. Nous simplifierons par la suite l'expression pour calculer séparément l'état de chaque voxel et diminuer la complexité du calcul, mais il est nécessaire dans un premier temps de travailler avec l'état global de la grille pour permettre d'exprimer la prédiction de mouvement dans celle-ci.

Question à poser au modèle

Nous nous plaçons dans un contexte de *filtrage* temporel, qui est adapté par nature à notre approche temps réel de la reconstruction, effectuée en ligne. C'est à dire que notre problème d'estimation repose sur l'ensemble des observations présentes et passées, par opposition au *lissage* qui considère aussi les observations futures et donc la connaissance préalable de la séquence complète. Nous cherchons donc à inférer successivement à chacun des instants $t \in \{0, \cdots, T\}$, la distribution de probabilités des occupations sachant toutes les observations présentes et passées :

$$p(\mathcal{G}^t|\mathcal{I}^{0:t}) \tag{4.7}$$

4.6.2 Hypothèse Markovienne et récursivité

Le problème tel qu'il est posé, est complexe à résoudre. Il nous faut simplifier la distribution conjointe des variables et en restreindre les dépendances. Une hypothèse de travail pratique et très souvent utilisée est de considérer qu'un état du système ne dépend que de l'état précédent (système de Markov d'ordre 1). En pratique cela supposera que l'on puisse extrapoler le prochain état de la grille au temps $t+1$, à partir du seul état de la grille estimé à l'instant t. Un tel schéma possède déjà un grand pouvoir d'expression mais peut aussi être trop restrictif. Nous réévaluerons sa pertinence au moment d'analyser nos résultats.

L'hypothèse en question permet une simplification considérable de la distribution conjointe :

$$p(\mathcal{G}^{0:T}\ \mathcal{I}^{0:T}) = p(\mathcal{G}^0)\, p(\mathcal{I}^0|\mathcal{G}^0) \prod_{t=1}^{T} p(\mathcal{G}^t|\mathcal{G}^{t-1})\, p(\mathcal{I}^t|\mathcal{G}^t) \tag{4.8}$$

où l'on retrouve la dépendance à l'unique état antérieur dans les termes $p(\mathcal{G}^t|\mathcal{G}^{t-1})$.

La forme de cette distribution conjointe a une conséquence tout à fait bienvenue pour notre problème et son expression sous forme de l'inférence (4.7) : elle permet d'exprimer la question posée à l'instant t en fonction de celle posée

à l'instant précédent $t-1$, dans une boucle de *prédiction/estimation*. La phase d'*estimation* prend la forme suivante après application de la règle de Bayes :

$$p(\mathcal{G}^t|\mathcal{I}^{0:t}) = \frac{1}{\alpha}\, p(\mathcal{I}^t|\mathcal{G}^t)\, p(\mathcal{G}^t|\mathcal{I}^{0:t-1}) \tag{4.9}$$

Moyennant un facteur de normalisation α qui permet la sommation à 1 de la distribution, elle consiste à estimer la distribution recherchée en confrontant la dernière observation capteur (terme $p(\mathcal{I}^t|\mathcal{G}^t)$ qui correspond à la solution du problème statique) à un terme de *prédiction* $p(\mathcal{G}^t|\mathcal{I}^{0:t-1})$. Le rôle de ce terme de prédiction consiste, comme son nom l'indique, à prédire l'état de la grille en fonction des observations passées.

En développant l'expression de l'inférence de $p(\mathcal{G}^t|\mathcal{I}^{0:t-1})$, qui fait intervenir la distribution conjointe, il est possible d'exprimer ce terme de prédiction comme suit :

$$p(\mathcal{G}^t|\mathcal{I}^{0:t-1}) = \int_{\mathcal{G}^{t-1}} \left(p(\mathcal{G}^t|\mathcal{G}^{t-1})\, p(\mathcal{G}^{t-1}|\mathcal{I}^{0:t-1})\right) \tag{4.10}$$

Cette formule de prédiction admet une interprétation simple : moyennant le résultat de l'inférence au pas de temps précédent (terme $p(\mathcal{G}^{t-1}|\mathcal{I}^{0:t-1})$), ainsi qu'un modèle statistique de propagation d'un état $\mathcal{G}^{t-1}$ vers un état $\mathcal{G}^t$ (terme $p(\mathcal{G}^t|\mathcal{G}^{t-1})$) au cours d'un incrément de temps, la prédiction correspond à sommer les probabilités de tous les précédents états possibles de la grille, chacun pouvant contribuer au nouvel état $\mathcal{G}^t$ dans les proportions dictées par le modèle de propagation.

Ces résultats, relativement classiques, sont de portée générale et ne reposent que sur l'hypothèse d'un système Markovien d'ordre 1. Ils ne font pour l'instant intervenir aucune hypothèse propre à notre problème. L'aspect éminemment pratique de ceux-ci réside dans la possibilité de réutiliser de proche en proche le résultat de l'inférence au pas de temps précédent : on parle de *filtrage récursif*. Une spécialisation classique de cette formulation est le filtrage de Kalman [56].

En outre les solutions dans ce cadre ne reposent que sur la résolution séparée du problème statique $p(\mathcal{I}^t|\mathcal{G}^t)$ et sur la définition d'un modèle de propagation $p(\mathcal{G}^t|\mathcal{G}^{t-1})$. Du choix de ce dernier dépend l'efficacité de la méthode : il détermine la validité de la prédiction, qui doit être bonne et conduire l'estimation dans la(les) meilleure(s) direction(s) possible(s) pour ne pas perdre la "cible" ; sa forme et son acuité permettrons de ne sommer que sur un petit sous-ensemble de termes pertinents dans l'expression (4.10), déterminant donc également la calculabilité de la méthode. Nous expliquons comment parvenir à un tel but, en exposant la simplification de cette formulation proposée par la communauté de robotique dans le cadre des grilles d'occupation [27]. Nous affecterons ensuite une forme préliminaire au terme de propagation dans le cadre de notre problème.

4.6.3 Simplification pour les grilles d'occupation

Les expressions précédemment données couvrent le problème du filtrage dans sa plus grande généralité. Il permet d'exprimer l'évolution de la grille dans son ensemble. Même si un tel calcul peut avoir un sens, il reste d'une grande complexité et peu adapté au temps réel. Une simplification a été proposée dans [27] pour effectuer l'estimation de chaque voxel de la grille indépendamment. Elle consiste à reprendre les expressions (4.9) et (4.10) et à les reformuler pour le cas d'un seul voxel X, en posant la question de sa probabilité d'occupation sachant les observations à chaque instant t. La phase d'estimation s'exprime alors de la manière suivante :

$$p(\mathcal{G}_X^t|\mathcal{I}^{0:T}) = \frac{1}{\alpha}\, p(\mathcal{I}^t|\mathcal{G}_X^t)\, p(\mathcal{G}_X^t|\mathcal{I}^{0:t-1}) \tag{4.11}$$

où $p(\mathcal{I}^t|\mathcal{G}_X^t)$ est le modèle capteur du cas statique précédemment traité. Il est proposé d'écrire la phase de prédiction pour un voxel sous la forme suivante :

$$p(\mathcal{G}_X^t|\mathcal{I}^{0:t-1}) = \int_{X^{t-1}} \left(p(X^t|X^{t-1})\, p(\mathcal{G}_X^{t-1}|\mathcal{I}^{0:t-1})\right) \tag{4.12}$$

La prédiction consiste alors à sommer les probabilités de tous les voxels de la grille estimés à l'instant précédent, chacun pouvant contribuer au nouvel état du voxel $\mathcal{G}_X^t$ dans les proportions dictées par le modèle de propagation $p(X^t|X^{t-1})$. Ce dernier régit les possibilités de déplacement de la matière d'un voxel de l'instant précédent.

Une simplification supplémentaire est suggérée pour assurer la faisabilité du calcul pour ce modèle : réduire la sommation de l'expression (4.12) à un ensemble restreint L_{X^t} de voxels de l'instant $t-1$, obtenus par tirage de la distribution de propagation $p(X^t|X^{t-1})$:

$$p(\mathcal{G}_X^t|\mathcal{I}^{0:t-1}) \approx \frac{1}{|L_{X^t}|} \sum_{X^{t-1}\in L_{X^t}} p(X^t|X^{t-1})\, p(\mathcal{G}_X^{t-1}|\mathcal{I}^{0:t-1}) \tag{4.13}$$

Reste donc, dans le cadre de ces simplifications, à définir le modèle local de propagation $p(X^t|X^{t-1})$. Celui proposé dans le cadre des travaux [27] est spécifique au cas de capteurs sonars traquant des cibles ponctuelles et à des situations rencontrées spécifiquement en robotique, où l'état prend en compte des observations de la vitesse des cibles. Travaillant avec des capteurs et des conditions de nature très différentes, il nous faut proposer un modèle de propagation spécifique, ce que nous faisons au prochain paragraphe.

4.6.4 Prédiction isotrope de mouvement

Nous présentons ici un modèle très simple qui permet de se rendre compte de la nature de l'information que permet d'exploiter le filtrage. Il s'agit d'une solution préliminaire aux possibilités relativement limitées : nous discuterons à la fin du chapitre des extensions possibles qui peuvent l'améliorer.

En l'absence d'information supplémentaire sur la vitesse de la matière, que nos caméras ne donnent pas sans envisager des traitements complexes additionnels, nous pouvons simplement supposer que la matière d'un voxel peut se déplacer de manière équiprobable d'un voxel soit vers lui-même, soit vers un de ses 26 voxels voisins. Ce modèle donne une forme triviale à $p(X^t|X^{t-1})$: il est très simple à calculer et permet d'accumuler une information temporelle principalement pour les mouvements lents d'objets larges dans la scène. Il filtre en effet par nature tout mouvement ample et rapide susceptible de produire un déplacement sur plus d'un voxel de distance d'un pas de temps au suivant. C'est pourquoi nous proposons des expérimentations dans le cadre de la localisation et du suivi d'objet plutôt que dans un contexte de modélisation de surface, où cette forme de prédiction n'a pas vraiment de sens.

4.6.5 Résultats

Nous reprenons, pour ces résultats, la séquence multi-objet se déroulant dans un large espace d'acquisition. La figure 4.8 montre, pour quelques instants de la séquence, une coupe horizontale de la grille d'occupation calculée, avec les résultats du schéma de prédiction présenté, ceux de l'estimation, ainsi que les résultats purement statiques obtenus à titre de comparaison. Les résultats sont obtenus avec les même réglages des paramètres que pour les résultats statiques obtenus dans les paragraphes précédents.

Au tout premier instant nous avons naturellement donné une forme uniforme à la prédiction, occasionnant une estimation aux résultats parfaitement identiques à ceux obtenus dans un cadre statique. Ensuite, comme le montrent les coupes des instants 2,5 et 10, le filtrage provoque une propagation rapide d'information dans les premiers instants de la séquence. Les endroits vides sont rapidement identifiés et chaque pas supplémentaire dans le calcul corrobore un peu plus l'absence de matière, sauf dans la zone supérieure gauche, vue par très peu de caméras, et la zone inférieure gauche, où une personne est en train de marcher. Entre l'instant 15 et 70, la personne passe dans une zone couverte par seulement trois caméras (en bas à droite) et est donc presque perdue, comme le montrent les coupes de l'instant 20. Le calcul statique permet de localiser la cible pendant quelques instants de plus que la version filtrée. La cible réapparaît un peu avant l'instant 80, la personne marchant dans une zone vue par plus de caméras et donc plus fiable. Elle est alors rejointe pour le reste de la séquence par une seconde personne

venant de la zone supérieure gauche de la scène. Dans les coupes 132 et 160 les deux personnes marchent dans l'aire d'acquisition. Les coupes 80, 132 et 160 font apparaître un nombre de zones probables supérieur à 2. De larges zones dans les régions périphériques de l'espace d'acquisition sont rendues aussi probables que les véritables cibles. Ceci est dû à une conjonction de phénomènes malheureusement courants dans une situation où l'on cherche à inférer de l'information sur la scène à partir de silhouettes. Les régions périphériques sont vues par peu de caméras, ce qui rend leur observation peu fiable. De plus, les seules caméras voyant ces zones voient également une cible : les régions périphériques rendues probables sont dans l'ombre d'une cible véritable, ce qui créé des zones fantômes et donc de fausses cibles. L'erreur commise dans ces zones est également cohérente dans le temps, et le filtrage conforte donc le système dans son erreur.

Les grilles statiques sont donc généralement plus fiables que celles calculées par le filtrage proposé. Nous pouvons voir, dans les cas statiques, que les zones périphériques observées par peu de caméras présentent également des régions ambiguës dans l'ombre de véritables cibles. Cependant, au cours de l'estimation statique, il est mathématiquement impossible que celles-ci soient plus probable qu'une cible véritable observée par un nombre supérieur de caméras. Ceci n'est plus vrai dans le cadre du filtrage proposé, où autant d'importance est accordée à la prédiction (et donc aux observations passées corrigées par le modèle de mouvement proposé) qu'aux observations présentes. L'estimation statique permet donc une localisation et une séparation des cibles par rapport au bruit beaucoup plus fiable qu'avec le filtrage trop simpliste proposé.

Ces résultats font apparaître l'importance du choix de la fonction de propagation utilisée pour la prédiction. Une meilleure prédiction ferait probablement intervenir un champ de vitesse calculé au niveau de la grille, pour un suivi volumique beaucoup plus précis. Cependant, à moins d'utiliser une prédiction basée sur un modèle de plus haut niveau, la prédiction risque toujours de se faire piéger dans les zones périphériques où le système corrobore ses propres erreurs, à cause de l'ambiguïté de l'information fournie par les silhouettes. Nous pensons donc qu'avant toute proposition plus élaborée pour la fonction de propagation, il faut tempérer davantage les observations effectuées dans les zones observées par peu de caméras, en introduisant dans le problème un *a priori* supplémentaire par rapport au nombre de caméras voyant un voxel. Il faut, selon nous, pouvoir corréler plus fortement la fiabilité d'inférence sur un voxel avec le nombre d'observations dont on dispose de celui-ci. Ce nombre est en effet connu à partir du moment où l'on suppose connue la pose des caméras. Notons que cet *a priori* supplémentaire est inutile dans le cas statique où le système réalisé tempère déjà l'information disponible aux endroits peu observés.

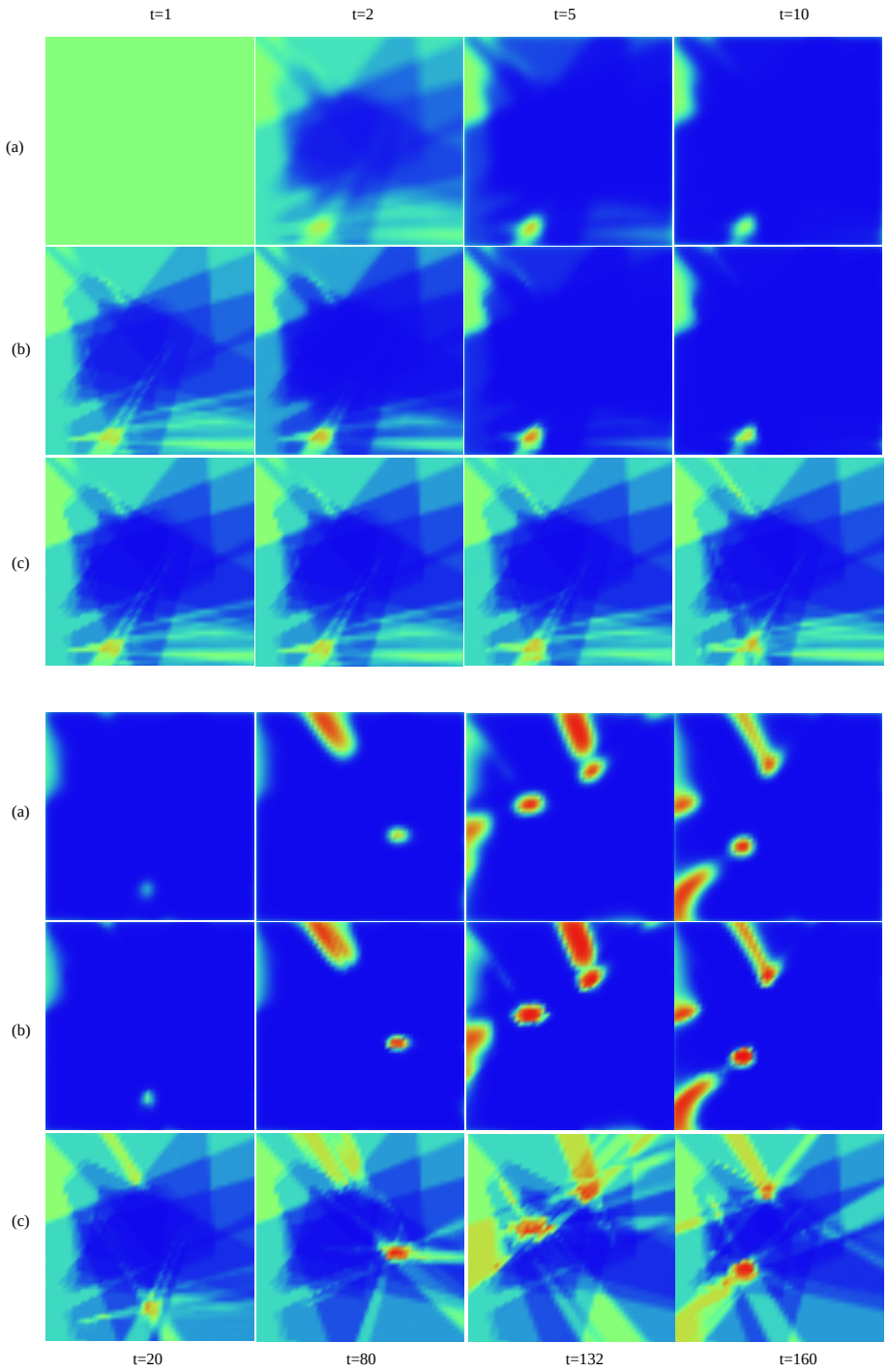


FIG. 4.8 – Vue en coupe horizontale de la grille d'occupation calculée pour les instants 1,2,5,10 (haut) et 20,80,132,160 (bas). (a) résultat de la prédiction. (b) résultat de l'estimation. (c) résultat du calcul statique pour comparaison.

4.7 Discussion

Nous avons présenté une nouvelle approche pour la fusion d'information silhouette provenant de plusieurs vues. Nous utilisons une approche rigoureuse de fusion de capteurs, pour lier directement l'information de la scène aux observations. Ceci présente plusieurs avantages. Tout d'abord la chaîne entière, des causes aux observations, est modélisée. Toutes les hypothèses faites sont rendues explicites. La méthode évite aussi par conséquent de prendre des décisions prématurées sur la segmentation des silhouettes, ce qui aurait requis l'ajustement manuel de paramètres pour chaque image. Ainsi, toute l'information silhouette disponible peut être intégrée, en utilisant seulement trois paramètres globaux du modèle capteur. Ces paramètres contrôlent intuitivement la certitude que l'on accorde aux observations.

Nous avons en outre effectué une analyse préliminaire d'un tel système dans un cadre dynamique, en optant pour un filtrage récursif de la grille et une fonction de prédiction simple. Cette étude montre le potentiel du modèle proposé dans l'intégration d'informations de cohérence temporelle, et montre aussi les difficultés liées à la structure de notre problème. Les informations image et le raisonnement sur les silhouettes peuvent en effet conduire le système à des conclusions ambiguës aux endroits vus par peu de caméras.

L'approche proposée a été validée dans le cadre de diverses applications, et de nombreuses nouvelles idées peuvent être essayées et rajoutées sans changer le cœur de la méthode. Nous pourrions, en l'occurrence, prendre en compte plus de dépendances statistiques dans le modèle. Notamment, nous avons remarqué que la fiabilité des observations d'un pixel peut être liée à la couleur observée à ce pixel. Par exemple nous observons de nombreuses fois le cas ou un objet d'intérêt noir apparaît devant un fond noir, qui engendre spécifiquement des erreurs. Ceci pourrait être l'objet d'un traitement particulier et pourrait être examiné. D'autres informations *a priori* peuvent être intégrées, pour tempérer l'inférence en fonction du nombre de caméras qui observent une zone, puisque l'on a pu observer expérimentalement la baisse de fiabilité du système dans ces zones dans le cadre du problème dynamique.

Néanmoins l'approche fournie est déjà fonctionnelle et fournit des résultats utilisables telle que présentée dans sa formulation statique. Nous sommes donc confiants dans le fait que la méthode proposée ouvre des perspectives intéressantes et fera l'objet de travaux futurs.

5.

Applications temps réel

Nous avons proposé plusieurs algorithmes efficaces pour traiter l'information provenant des silhouettes, que celles-ci soient représentées de manière statistique ou purement géométrique. Nous nous penchons dans ce chapitre sur le cas d'étude de la reconstruction en temps réel dans un studio virtuel comportant plusieurs caméras et un grand écran pour la visualisation. Un tel studio présente de nombreuses possibilités d'interaction pour la réalité virtuelle et mixte, en se basant sur la reconstruction de la forme d'un ou plusieurs utilisateurs et d'objets présents dans la scène : on parle alors de *virtualisation* de la scène.

Obtenir de telles formes à partir de silhouettes avec les algorithmes présentés nécessite d'utiliser environ une dizaine de caméras, ou plus, selon les applications. Mais le calcul de forme pour 10 vues en temps réel est déjà lourd et nécessite de manipuler des flux de données trop importants pour une seule machine. De plus il est généralement nécessaire pour des raisons matérielles d'utiliser une machine par caméra pilotée, qui se charge de l'acquisition et de certains prétraitements. C'est donc naturellement que nous nous tournons vers un traitement distribué des tâches, propice à utiliser le matériel informatique déjà disponible et à permettre au système de passer à l'échelle, lorsque l'on souhaite lui ajouter des vues pour la reconstruction, ou lui adjoindre d'autres fonctionnalités.

Nous présentons donc dans ce contexte une plate-forme d'acquisition et d'interaction, la plate-forme GrImage. La configuration proposée est composée d'une salle équipée de caméras, d'une grappe d'ordinateurs pour les piloter et mettre en œuvre l'interactivité du système, et d'un écran haute résolution pour le retour utilisateur. Nous proposons pour ce type de configuration un algorithme de reconstruction géométrique distribué. Celui-ci est issu d'une réflexion sur les stratégies de distribution à mettre en oeuvre pour tirer parti de ce type d'environnement, que nous détaillons. Une application d'interaction complète s'appuyant sur la modélisation à partir de silhouettes, et l'architecture associée, sont proposées et analysées au cours de ce chapitre.

5.1 Introduction

Les évolutions technologiques récentes des caméras (taux de transferts plus rapides, normalisation des interfaces, augmentation des fréquences d'acquisitions) ont permis une large diffusion de l'acquisition temps réel d'images numériques. Ceci permet de concevoir des systèmes complets d'acquisition en connectant simplement un ensemble de caméras et de PCs sans avoir recours à du matériel spécifique. De nombreux efforts de recherche s'intéressent à des configurations comportant un petit nombre de caméras connectées à un seul, ou à un petit nombre d'ordinateurs. Très peu de travaux s'intéressent aux situations où un plus grand nombre de caméras et PCs sont impliqués, environnements qui deviennent pourtant de plus en plus courants et peu onéreux. De plus, la plupart des applications de vision utilisant plusieurs caméras connectées à plusieurs PCs n'utilisent pas pleinement la puissance de calcul disponible et reposent généralement sur un calcul séquentiel sur une seule machine. Or cette puissance de calcul peut être utile dans notre problème : bien que rapides, les algorithmes que nous proposons ne permettent pas de réaliser un système temps réel dur (30Hz *garantis*) pour une reconstruction à partir d'un nombre de vues important. Par conséquent nous nous orientons naturellement ici vers une solution distribuée permettant d'utiliser pleinement les ressources disponibles. Nous nous intéresserons donc aux problèmes liés à la nature interactive du système souhaité : les problèmes de passage à l'échelle pour un nombre élevé de caméras, le choix de stratégies de parallélisation pour la modélisation 3D, le contrôle des flux de données. Nous souhaitons montrer que ce cadre de travail rend réalisable la création d'un système de virtualisation interactif, utilisant la modélisation 3D à partir de silhouettes.

5.1.1 Systèmes existants

Seul un petit nombre de systèmes distribués ont déjà été élaborés pour la modélisation 3D. L'institut de robotique CMU propose l'utilisation d'un dôme constitué d'une cinquantaine de caméras pour la virtualisation [80] ; le modèle 3D de la scène est obtenu via une approche stéréoscopique. D'autres systèmes ont aussi été proposés à CMU avec moins de caméras. L'un est basé sur une modélisation volumétrique de l'enveloppe visuelle [22] et propose un traitement temps réel, sans se pencher sur le parallélisme étant donné qu'un faible nombre de caméras est utilisé. L'autre propose d'effectuer une modélisation volumétrique en exploitant la cohérence temporelle grâce à un alignement de nature photométrique [21]. Cependant, ce dernier système n'est conçu que pour un traitement hors-ligne.

Une autre catégorie d'approches temps réel mais non parallèles utilise les processeurs graphiques dédiés pour créer directement des images issues de nouveaux points de vue [68]. L'utilisation des cartes graphiques permet de grandement accélérer le rendu mais de tels systèmes reposent encore sur un seul ordinateur

pour les calculs, et ne fournissent pas les modèles 3D explicites requis par de nombreuses applications pour tout post-traitement avancé du modèle.

Davis *et al.* [14] présentent un système distribué pour l'enregistrement, l'accès et le traitement de flux vidéo importants sur une grappe de PC. Mais ces travaux mettent l'accent sur l'aspect base de données et ne considèrent pas les applications temps réel.

Des systèmes de modélisation parallèles et temps réel ont été proposés pour gérer la modélisation volumétrique [57, 8]. De tels calculs peuvent être facilement distribués étant donné qu'une part importante des calculs est effectuée indépendamment pour chaque voxel. Les approches mentionnées utilisent cette propriété et obtiennent de bonnes performances, mais n'abordent que succinctement les problèmes difficiles de parallélisme du fait de la trivialité des traitements dans ce cas particulier. De plus, nous avons déjà évoqué le mauvais rapport entre le coût de calcul et la précision atteinte par les approches volumétriques, c'est pourquoi nous souhaitons explorer les possibilités de parallélisation d'algorithmes géométriques offrant un autre compromis, basés sur le calcul de surfaces. Nous tentons par la même occasion de prendre davantage de recul par rapport au problème du traitement parallèle en vision pour permettre au lecteur d'appliquer les concepts présentés à d'autres problèmes similaires.

Plusieurs travaux intéressants concernent cette problématique de parallélisation pour la vision par ordinateur. Medioni *et al.* [37] proposent une stratégie multi-threads pour améliorer la latence et le débit des traitements vidéo tels que la segmentation ou le suivi de cible. Le projet Skipper [91] fournit un environnement de programmation parallèle pour le traitement d'images. Cependant ces travaux ne fournissent principalement qu'un environnement de conception parallèle adapté, mais ne considèrent pas les aspects algorithmiques pourtant vitaux pour l'application de modélisation visée.

5.1.2 Contributions

Dans ce chapitre, nous présentons une architecture parallèle et temps réel pour les algorithmes multi-caméras et son application à la modélisation 3D à partir de silhouettes. Notre but est de fournir des solutions pouvant passer à l'échelle, c'est à dire ne s'écroulant pas avec l'adjonction de caméras et données d'entrée supplémentaires, en utilisant une stratégie de parallélisation. Cette stratégie est conçue pour être suffisamment générale pour s'appliquer à différents contextes tout en limitant l'effort de parallélisation. Notre contribution par rapport aux travaux existants est double : d'une part nous étendons aux algorithmes multi-caméras les concepts de parallélisation déjà proposés dans la littérature, d'autre part nous montrons l'application de ces concepts à la modélisation polyédrique exacte à partir de silhouettes, et proposons une solution pratique, efficace, et passant à l'échelle pour un grand nombre de vues.

L'organisation du chapitre est la suivante. Le paragraphe 5.2 introduit les concepts et stratégies utiles pour la distribution des traitements dans le cadre de notre problème. Son application au calcul de l'enveloppe visuelle à partir d'images est présentée au paragraphe 5.3. Le paragraphe 5.4 valide les principes proposés dans un contexte réel et fournit des preuves numériques à partir de données synthétiques.

Ces travaux ont fait l'objet de publications en collaboration avec Clément Ménier, Jérémie Allard, Edmond Boyer et Bruno Raffin [76, 42, 77, 79]. Les expérimentations ont lieu sur la plate-forme d'acquisition GrImage [79], fruit d'un effort collaboratif de nombreuses équipes de recherche et personnels de l'INRIA Rhône-Alpes.

5.2 Stratégie de distribution

Le traitement multi-vue et la reconstruction de formes à partir de silhouettes sont des tâches très particulières. Pour proposer un algorithme distribué efficace, nous nous penchons ici sur les techniques existantes de traitement parallèle, et dégageons les notions et concepts utiles pour nos objectifs. Il s'agit bien sûr d'identifier les caractéristiques d'un algorithme parallèle, son surcoût par rapport à un traitement séquentiel, et la nature des ajustements possibles d'un tel algorithme pour amortir ce surcoût et en tirer un bénéfice. Mais il s'agit aussi de comprendre comment créer des algorithmes distribués avec un effort humain raisonnable.

5.2.1 Définitions

Nous introduisons tout d'abord quelques définitions utiles. Dans ce contexte de parallélisme, nous désignerons par le mot *nœud* un ordinateur impliqué dans le traitement distribué. Nous supposons toujours disposer de n vues, et donc que n caméras sont physiquement utilisées sur la plate-forme. Nous disposons également de m nœuds pour le traitement.

Nous supposons que chaque caméra est connectée à un nœud consacré à l'acquisition et aux prétraitements de l'image. Dans le cas de notre algorithme ces prétraitements sont l'extraction de la silhouette, et sa vectorisation, ce qui permet de réduire la taille de l'information à transmettre aux autres nœuds. Nous considérons que toutes les caméras fournissent des images à la même fréquence : celle-ci détermine le *débit* des flux d'images qui entrent dans le système. Nous supposons donc naturellement que $m > n$; les $p = m - n$ nœuds restants sont consacrés à tout traitement autre que l'acquisition.

Nous appellerons *trame* l'ensemble des n images prises à un instant t. Les nœuds sont supposés inter-connectés via un réseau standard. Accéder à une donnée

située sur un nœud distant est donc beaucoup plus lent qu'accéder à une donnée locale. Il s'agit là d'une hypothèse importante : l'architecture visée ne comporte aucune mémoire partagée, et il est préférable de n'utiliser aucun outil pour simuler virtuellement un tel espace mémoire. Nous préconisons plutôt de gérer explicitement les transferts de données, comme le permettent des outils comme MPI [48]. Le contrôle et l'optimisation des flux pouvant être mis en œuvre alors sont en effet primordiaux dans une telle architecture, et sont une des clés de la performance du système. Comme nous le verrons par la suite, le traitement spécifique des transferts de données induit un effort relativement faible pour un gain significatif.

Nous mesurons, en tant que critère de performance, l'*accélération* (en anglais *speedup*), obtenue en divisant le temps d'exécution séquentiel par le temps d'exécution parallèle. L'efficacité de la parallélisation est d'autant plus grande que le facteur d'accélération est proche du nombre de processeurs. Pour les contraintes de temps réel, nous mesurons bien sûr le débit, mais aussi la *latence*, c'est à dire la quantité de temps incompressible nécessaire pour le traitement d'une unique trame, qui détermine la réactivité du système. Dans le cas d'un algorithme de reconstruction parallèle, il s'agit du temps écoulé entre la date de réception des images issues d'une acquisition (les données d'entrées pour une trame), et la production du modèle associé à cette trame. Comme nous le détaillerons plus loin la latence n'est pas simplement l'inverse du débit comme c'est le cas pour les traitements séquentiels, puisque plusieurs trames peuvent être simultanément en cours de traitement.

Nous proposons un formalisme de parallélisation simplifié, adapté aux problèmes de vision multi-caméra que nous voulons traiter. Celui-ci constitué de deux niveaux, offrant deux axes d'ajustement du débit et de la latence : la parallélisation *par flux* et la parallélisation *par trame*.

5.2.2 Parallélisation par flux

Nos applications multi-caméras traitent une séquence de trames, c'est à dire un *flux* multi-images. Une idée classique pour accélérer des applications traitant des flux est d'utiliser un *pipeline*. L'application est alors scindée en une séquence d'étapes (cf. fig. 5.1(b)), la responsabilité du traitement de chaque étape étant attribuée à différents nœuds consacrés. Ceci permet à un nœud associé à une étape de l'algorithme de traiter la trame t, pendant que le nœud chargé de l'étape suivante du pipeline traite la trame $t-1$. Les différentes étapes sont en général naturellement déduites de la structure de l'algorithme à paralléliser. Il est généralement inefficace de chercher à augmenter artificiellement le nombre d'étapes d'un algorithme pour obtenir en un pipeline plus long. En effet le coût d'une telle mise en œuvre peut être élevé sur une grappe de calcul, et les performances générales de l'algorithme peuvent diminuer, en particulier si ce remaniement nécessite de communiquer sur le réseau des quantités importantes de données intermédiaires entre

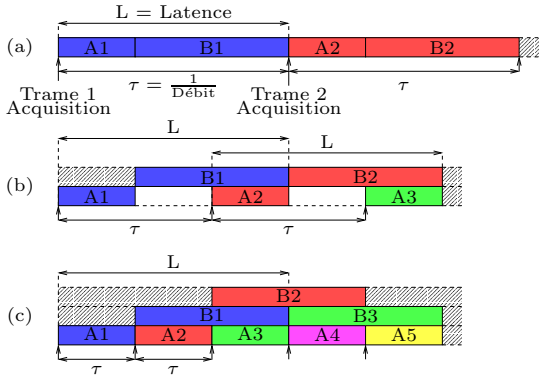


FIG. 5.1 – Parallélisation par flux. Supposons que A et B sont les deux étapes de calcul de notre programme. At et Bt correspondent au traitement de la trame t. Chaque ligne correspond à un processeur. Les blocs colorés correspondent à l'exécution d'une tâche et les blocs blancs à des périodes d'inactivité. Le schéma (a) représente une exécution purement séquentielle, le schéma (b) une exécution avec un pipeline à 2 étapes. Le schéma (c) ajoute une seconde unité de calcul pour la seconde étape B du pipeline.

les étapes créées. Dans ce cas le temps passé à transmettre des données augmente et avec lui la latence de l'algorithme.

Une étape est dite *statiquement calculable* si elle n'utilise pas de cohérence temporelle, c'est à dire qu'elle traite la trame t indépendamment des informations liées aux trames autres que t. Cette propriété est intéressante et permet d'utiliser plusieurs nœuds pour le traitement de l'étape (voir figure 5.1(c)). On parle alors d'un ensemble d'*unités de calcul* d'une même étape : une nouvelle trame t peut être traitée dès qu'une des unités de calcul de l'étape est disponible. Le nombre d'unités de calcul devrait être suffisamment grand pour éviter toute attente de traitement d'une trame, et doit donc être ajusté selon la durée du traitement et sa proportion par rapport aux autres étapes. Adapter cette technique à une étape non statiquement calculable peut s'avérer réalisable mais nécessite des techniques d'ordonnancement avancées et des communications supplémentaires non détaillées ici.

5.2.3 Parallélisation par trame

Les techniques de distribution précédentes peuvent améliorer significativement le débit, en permettant d'augmenter le nombre de trames traitées simultanément. Cependant, elles introduisent une latence supplémentaire, du fait des communications réseaux introduites entre les nœuds. Pour améliorer la réactivité du système et donc diminuer la latence, il nous faut réduire le temps global de traitement d'une trame. Ceci nécessite de distribuer le travail effectué *au sein d'une même*

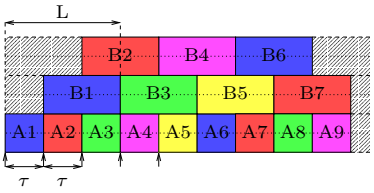


FIG. 5.2 – Parallélisation par trame (deux processeurs pour l'étape A et quatre processeurs pour l'étape B) : amélioration de la latence par rapport à la parallélisation par flux (cf. fig. 5.1.)

trame. Nous introduisons donc un niveau de granularité supplémentaire dans l'architecture : non seulement le pipeline se subdivise en unités de calculs qui sont affectés à un étage du traitement pour une seule trame donnée, mais l'unité de calcul elle-même peut comprendre plusieurs nœuds, pour le traitement parallèle d'une étape à une trame donnée (voir figure 5.2).

Il nous faut donc maintenant réfléchir à la manière de mettre en œuvre du parallélisme sur l'architecture proposée. Nous proposons une approche basée sur le modèle classique du *Bulk Synchronous Programming* [102] qui propose un compromis entre performances et complexité d'implémentation. Ce modèle simplifie la description d'algorithmes parallèles en décomposant le traitement en phases de communications et de calculs. S'inspirant de ce modèle général, nous proposons un schéma simplifié plus spécifique, pour paralléliser les opérations au sein d'une unité de calcul. Ce schéma s'articule en trois phases :

- **Préparation des données** : cette phase consiste à envoyer les données d'entrée nécessaires aux nœuds concernés. Chaque nœud effectue alors localement les opérations d'initialisation requises par les phases de traitements suivantes.
- **Calcul parallèle** : en parallèle, chaque nœud exécute localement (sans communications) une tâche différente qui lui a été assignée.
- **Calcul séquentiel** : les résultats partiels issus de la phase précédente sont réunis sur un seul nœud. Ce nœud effectue de manière séquentielle le reste des calculs qui n'ont pu être effectués lors de la phase parallèle. Selon la nature des données qui doivent transiter entre cette étape et la suivante, il peut être plus efficace de dupliquer ce calcul séquentiel sur plusieurs nœuds (et donc de fusionner conceptuellement le calcul séquentiel avec la préparation de données et l'initialisation de l'étape suivante). Ceci est rentable si le temps passé sur ces calculs redondants permet d'économiser des communications, en faisant transiter des données de taille moins conséquente vers l'étape suivante.

Bien que très simple ce modèle s'avère très général. Dans le pire des cas, tous les calculs sont effectués dans la dernière phase séquentielle. Cependant cela est

évidemment inefficace. Nous montrons par la suite qu'il est possible d'obtenir une phase de calcul parallèle significativement plus importante que les deux autres phases. Dans de telles situations, nous montrerons que ces concepts permettent d'atteindre des performances temps réel pour notre problème de modélisation.

A titre d'exemple, nous illustrons l'expressivité du traitement proposé sur la méthode volumétrique décrite par Arita *et al.* [8], dont il est facile d'identifier les phases de parallélisation. Il s'agit d'un algorithme en une étape, avec les phases suivantes :

- Préparation des données : initialisation de l'espace des voxels.
- Calcul parallèle : pour chaque image, calcul du cône visuel dans l'espace des voxels.
- Calcul séquentiel : rassemblement des cônes visuels sur un nœud et calcul de leur intersection.

Les résultats présentés par Arita *et al.* montrent de bonnes performances. Généralement, le traitement en trois phases proposé ne fournit pas une parallélisation optimale. D'autres optimisations peuvent encore être apportées. Par exemple, pour calculer la même enveloppe visuelle volumétrique, Borovikov *et al.* ont décrit une optimisation algorithmique [14] pour le calcul de l'intersection des cônes visuels d'une manière plus complexe, non représentable en trois phases. Cependant nous montrons dans ce chapitre que la stratégie présentée offre une méthode simple et efficace pour la parallélisation avec des contraintes temps réel.

5.3 Reconstruction géométrique distribuée

Nous avons proposé au chapitre 3 deux méthodes pour obtenir une surface polyédrique de l'enveloppe visuelle : une méthode hybride basée sur la triangulation de Delaunay, étudiée au paragraphe 3.5, ainsi qu'une méthode polyédrique exacte, étudiée au paragraphe 3.7. Nous proposons ici une application des concepts du paragraphe précédent au calcul de l'enveloppe visuelle polyédrique exacte. A titre informatif il est aussi possible de distribuer l'algorithme hybride, ce qui peut s'avérer efficace si l'on calcul de manière parallèle la triangulation de Delaunay. Cette parallélisation a déjà été étudiée dans certains travaux [25], autour de l'idée d'une subdivision de l'espace en sous-régions dont le calcul peut s'effectuer de manière concurrente. Ces travaux mettent en lumière certains problèmes types pouvant se poser pour la distribution d'algorithmes géométriques. Il apparaît cependant que la triangulation de Delaunay est un algorithme intrinsèquement plus difficile à rendre parallèle du fait que la tétraédrisation à calculer est globalement contrainte : tout calcul local peut avoir des répercussions ou être en désaccord avec un autre calcul dans une région tout à fait différente. C'est pourquoi notre attention se porte

ici sur la parallélisation de l'algorithme polyédrique exact, qui ne fait intervenir que des calculs locaux et gloutons, et se prête donc plus facilement à la tâche.

Rappelons que l'algorithme de modélisation polyédrique exacte se déroule en trois étapes : le calcul des segments de vue, puis le calcul des arêtes d'intersection de cônes, enfin l'identification des contours et facettes du polyèdre. Étant donné que les flux de données d'une étape à l'autre de cet algorithme sont de taille raisonnable, nous proposons naturellement d'identifier ces trois étapes à trois étages d'un pipeline de traitement. A tout moment nous ne manipulons en effet que des maillages partiels du polyèdre dont la taille n'est pas de nature à écrouler le réseau de communication utilisé. Remarquons que toute étape de l'algorithme est statiquement calculable, ce qui nous permet d'utiliser plusieurs unités de calcul pour chaque étape. L'architecture proposée est schématiquement représentée en figure 5.3.

La création d'un pipeline permet d'augmenter le débit, mais pas la latence, c'est pourquoi nous nous penchons dans la suite de ce paragraphe à la parallélisation par trame associée à chacune de ces étapes. Il s'agit là de la partie la plus spécifique d'une parallélisation car elle nécessite de s'attaquer au fonctionnement propre de l'algorithme, avec des possibilités de choix ouvertes pour sa décomposition en sous-problèmes pouvant être résolus de manière concurrente. Nous formulons ci-dessous nos choix de parallélisation pour chaque étape du calcul de l'enveloppe visuelle exacte.

5.3.1 Étape des segments de vue

Cette étape donne une grande liberté pour le parallélisme, car elle consiste à calculer de nombreux résultats partiels et indépendants. En effet chaque ligne de vue peut être traitée indépendamment des autres, avec pour seul prérequis la disponibilité de toute l'information silhouette des images sur chaque nœud. Une stratégie efficace de parallélisation par trame peut donc être obtenue en partitionnant toutes les lignes de vue de toutes les images en p ensembles durant la phase de préparation des données, puis en distribuant chaque ensemble à l'un des p nœuds prévus pour le calcul pendant la phase de calcul parallèle. Il est important de bien équilibrer la charge entre les nœuds, pour éviter au maximum le temps passé à attendre le nœud le plus lent. Construire des ensembles de lignes de vue de cardinalité identique est une heuristique qui s'avère efficace. Notons que ce traitement impose la diffusion à tous les nœuds de toute l'information des contours des n silhouettes : il s'agit cependant d'un prérequis raisonnable dans la mesure où les silhouettes sont transmises sous forme polygonale et donc avec une taille faible. La finalisation de la tâche consiste simplement à réunir l'ensemble des résultats partiels sur chaque nœud concerné à l'étape suivante, pendant la phase de calcul séquentiel.

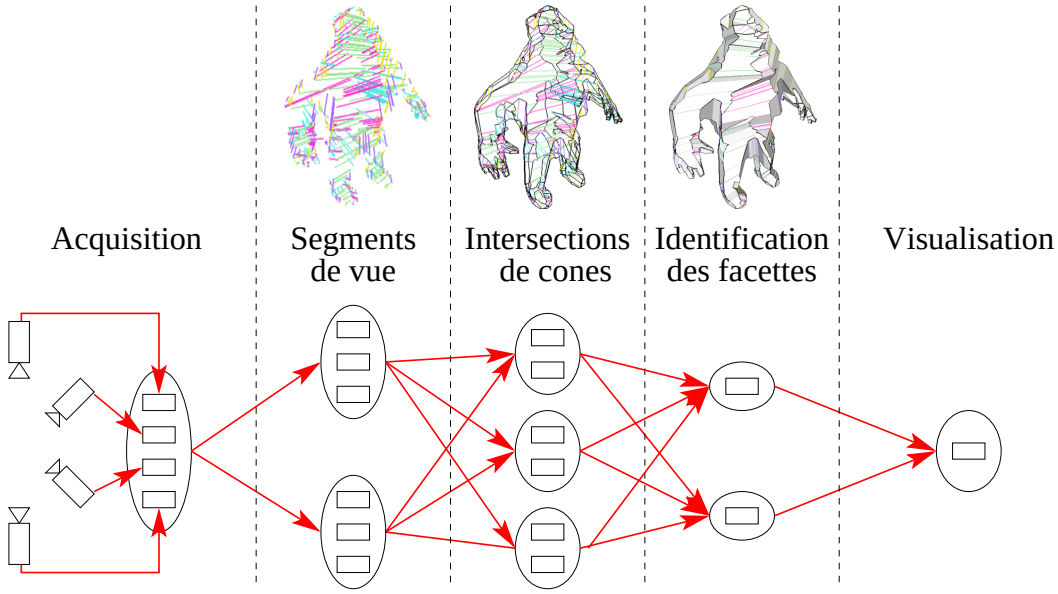


FIG. 5.3 – Architecture schématique proposée pour la distribution de l'algorithme de modélisation polyédrique exacte. Les ovales représentent les unités de calcul, les rectangles figurent les nœuds.

Grâce à cette stratégie de parallélisation nous atteignons des gains en vitesse d'un facteur 8 avec 10 nœuds pour cette étape de l'algorithme. C'est un très bon résultat, qui n'a nécessité qu'une adaptation succincte de l'algorithme séquentiel initial. Des accélérations encore meilleures peuvent être réalisées, mais au prix d'une complexité de mise en œuvre substantielle.

5.3.2 Étape du calcul d'intersections de cônes

Pour permettre une exécution concurrente, nous choisissons de manière classique de partitionner l'espace en p régions distinctes en utilisant $p-1$ plans parallèles, divisant ainsi l'espace en p "tranches". La largeur des tranches est ajustée en attribuant un même nombre de sommets de segments de vue à chaque tranche pour équilibrer la charge de calcul : il suffit de trier ces points selon l'axe de partition. Étant donné que nous ne manipulons que des sommets et des segments, le coût de partition de ces primitives durant la phase de préparation des données est relativement faible. Ainsi un nœud consacré à cette seconde étape de la méthode surfacique exacte peut être programmé pour calculer de proche en proche les arêtes d'intersection de cônes au sein de sa région désignée $\mathcal{R}_i$, jusqu'à ce qu'il rencontre des arêtes traversant les frontières vers une autre région $\mathcal{R}_j$. Ce nœud s'arrête alors de traiter cette courbe, déléguant le calcul du reste de la courbe au nœud en charge de la région $\mathcal{R}_j$. Cette stratégie est généralement efficace, mais toute arête

traversant une frontière entre régions sera calculée par deux nœuds différents. On ne peut en effet savoir si l'arête traverse une frontière qu'après l'avoir calculée. Une fusion minutieuse des arêtes redondantes est donc nécessaire à la fin de l'étape.

Il est assez aisé d'identifier les trois phases de la parallélisation par trame. La préparation des données consiste à partitionner l'espace en régions, et à distribuer les primitives géométriques concernées à chaque région ; la phase de calcul parallèle consiste à calculer les portions du maillage au sein de chaque région ; le calcul séquentiel réunit et fusionne les maillages partiels. Cette parallélisation s'avère efficace étant donné que nous obtenons un facteur d'accélération de 6 pour 10 nœuds avec notre implémentation, un bon résultat pénalisé principalement par les redondances de calculs aux frontières entre tranches. Les mesures globales fournies dans le paragraphe de résultats confirmeront l'intérêt de cette parallélisation, pour tout nombre de caméras.

5.3.3 Étape d'identification des facettes

Nous avons aussi distribué l'étape d'extraction de la surface, qui se prête bien au parallélisme : le maillage complet est diffusé à p nœuds durant la phase de préparation des données, ces p nœuds calculent ensuite un sous-ensemble de la surface (chaque facette du polyèdre peut être calculée de manière indépendante), et la phase séquentielle ne fait que collecter les différentes facettes calculées. Les accélérations obtenues sont très bonnes, de l'ordre de 7 pour 10 nœuds. Les seules pénalités viennent du fait que le traitement d'identification d'une facette est très simple et rapide (parcours de maillage), et que les communications réseaux prennent donc une part relative importante dans la latence.

5.4 Résultats

Nous avons réalisé un système expérimental comportant l'implémentation proposée de l'algorithme de reconstruction. Nous avons en outre effectué plusieurs manipulations permettant de se rendre compte des capacités du système pour l'interactivité, et des mesures quantifiant les débits, latences et la capacité de passage à l'échelle annoncée. Nous commençons donc par décrire le contexte expérimental choisi, c'est à dire la plate-forme GrImage.

5.4.1 Plate-forme GrImage

La plate-forme grimage est une plate-forme d'acquisition, de traitement et de visualisation dédiée à la réalité virtuelle et aux applications connexes. Elle comporte une dizaine de caméras de nature très différente (résolutions allant de 640×480 à 780×580), capables de fonctionner simultanément. L'ensemble des

calculs est effectué par une grappe de calcul hétéroclite, comportant des machines d'acquisition (Pentiums IV), des machines bi-processeurs (bi-Xeon 2.66GHz), et des machines 64 bits (opterons), cohabitant dans le même environnement de travail. L'ensemble de ces machines est relié par un réseau Gigabit, et huit d'entre elles, équipées de cartes graphiques puissantes comportant deux sorties graphiques (NVidia GeForce FX 6800), pilotent les 16 projecteurs de visualisation. Ceux-ci permettent un affichage sur un écran de 4×3 mètres avec une résolution de 4096×3072 pixels. La technologie actuelle des caméras vidéos présente des contraintes de fonctionnement : c'est pourquoi l'on propose sur cette plate-forme deux modes d'acquisition, synchrone ou asynchrone, en fonction des besoins. En mode asynchrone les images peuvent être acquises avec un débit de 30Hz mais peuvent comporter des écarts de datation de 33ms ; en mode synchrone les images sont synchronisées à l'ordre de la microseconde mais ne peuvent atteindre qu'un débit de 15Hz (mécanisme du Genlock). Chacun de ces modes comporte des avantages et inconvénients : l'absence de synchronisation favorise un meilleur débit, mais aussi des erreurs d'alignement entre vues qui engendrent des reconstructions pouvant être plus imprécises, allant jusqu'à la perte d'objets si ceux-ci se déplacent très rapidement. La synchronisation engendre pour l'instant des limitations de débit mais permet des reconstructions plus précises. Le mode asynchrone est cependant souvent suffisant si les mouvements capturés ne sont pas très rapides.

Les algorithmes décrits ont été implémentés sur la plateforme et permettent d'atteindre les conditions d'interactivité visées, c'est à dire des reconstructions à 30Hz, avec une latence d'environ 100ms, pour un nombre de caméras allant jusqu'à 12.

5.4.2 Validation pour un grand nombre de points de vue

Nous avons choisi de tester la capacité de passage à l'échelle de nos algorithmes distribués sur des images de plusieurs points de vue créées à partir d'un modèle synthétique, permettant une meilleure reproductibilité et flexibilité des conditions de mesure. Nous avons ainsi pu tester nos implémentations avec un nombre très important de caméras. Nous nous intéressons ici au problème de latence uniquement. Le problème du débit temps réel n'est pas discuté : il peut en effet être résolu facilement en multipliant le nombre de nœuds assignés à la parallélisation par flux. En pratique nos implémentations atteignent des débits de 30Hz en condition réelle de cette manière. Le débit du système n'est donc limité que par le débit des données transitant sur le réseau (qui dépend de la taille des données intermédiaires manipulées à chaque étape de l'algorithme) et la capacité de transmission physique de celui-ci.

Nous prenons pour nos expériences un modèle synthétique représentant un personnage humanoïde, présentant une complexité similaire à celle d'un person-

FIG. 5.4 – Modélisation et visualisation temps réel sur la plate-forme Grimage.

nage réel (sa projection dans les images engendre des silhouettes polygonales comportant environ 130 sommets). La figure 5.5 présente les latences obtenues avec différents nombres de processeurs utilisés pour 16, 25 et 64 points de vue. La parallélisation de l'algorithme permet de réduire la latence de manière significative (presque un ordre de grandeur). Avec 25 points de vue et 16 processeurs, la latence est inférieure à 200 ms, une latence acceptable pour l'interactivité du système.

La figure 5.6 présente les accélérations associées. Jusqu'à 9 processeurs pour 12 points de vue, 14 processeurs pour 25 points de vue, et plus de 16 processeurs pour 64 points de vue, le facteur d'accélération est supérieur à la moitié du nombre de processeurs utilisés. Au-delà, les facteurs d'accélération tendent à se stabiliser étant donné que la quantité de travail dans la phase de calcul parallèle diminue comparée à celle requise par les phases de préparation de données et de calcul séquentiel.

5.5 Discussion

Nous avons présenté un système de modélisation 3D qui utilise le parallélisme pour atteindre une exécution temps réel avec un nombre flexible de caméras et de PC. Un tel système repose sur un schéma de distribution que nous avons discuté et formalisé. Une telle stratégie de parallélisation peut être appliquée dans le

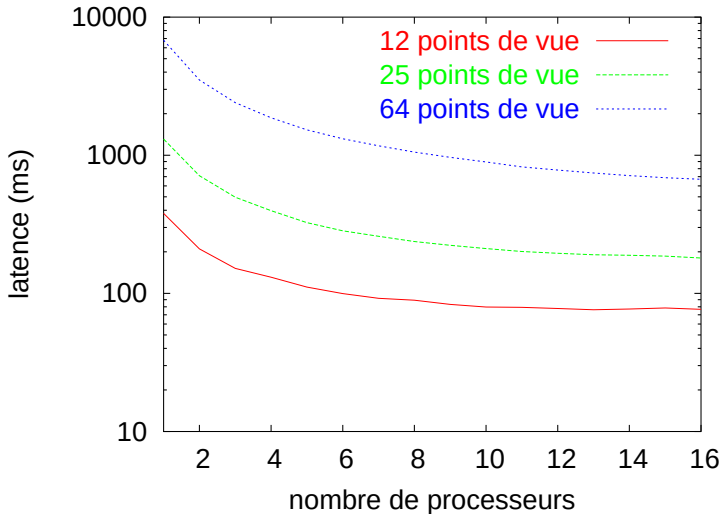


FIG. 5.5 – Graphe logarithmique des latences pour le personnage de synthèse.

contexte d'autres applications de vision manipulant plusieurs points de vue. Nous avons démontré son efficacité dans le cadre de la modélisation 3D à partir de silhouettes. Les enveloppes visuelles de haute qualité engendrées par ces algorithmes distribués peuvent être utilisées dans diverses applications, telles que la réalité virtuelle. Notre contribution principale par rapport aux travaux existants dans ce domaine consiste en la présentation de nouvelles implémentations parallélisées d'algorithmes de modélisation 3D ainsi que la formulation d'une stratégie de parallélisation pour les applications multi-vue. Des résultats sur des données réelles et de synthèse montrent que notre approche passe à l'échelle lors d'ajout de nouvelles vues et de nouveaux noeuds : il permet d'étendre le potentiel et la flexibilité de tels systèmes.

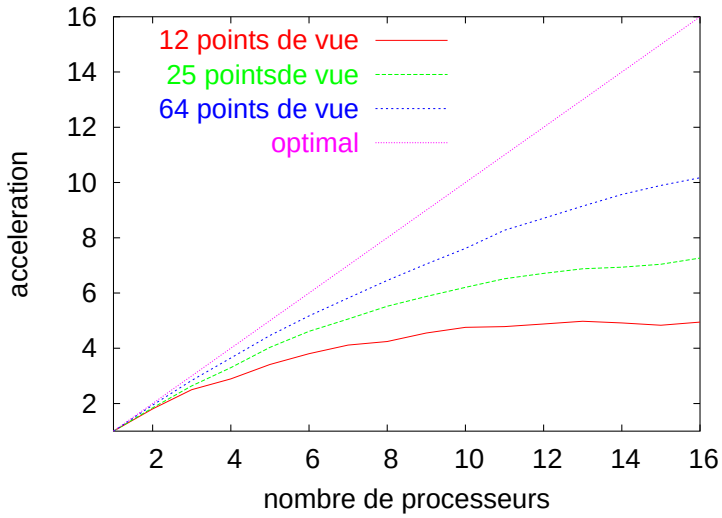


FIG. 5.6 – Accélérations pour le personnage de synthèse.

6.

Conclusion et perspectives

Nous avons traité dans cette thèse de modélisation de surfaces et de formes à partir de silhouettes, dans un contexte dynamique. Plusieurs problèmes fondamentaux ont été soulevés et étudiés, et diverses solutions proposées pour traiter le problème aussi bien d'une manière purement géométrique, qu'avec une méthodologie statistique. Enfin une réflexion sur la vision multi-vue dans un contexte distribué est fournie, ainsi qu'une solution distribuée de modélisation à partir de silhouettes.

Parmi les problèmes abordés et solutions proposées, cette thèse met en avant un certain nombre de résultats importants :

– Il est possible de construire des contours polygonaux à partir d'une silhouette binaire sans perte d'information. Ce résultat a un impact fondamental sur toute méthode se servant de silhouettes polygonales pour effectuer la modélisation, comme les deux algorithmes proposés, car elle permet de conserver une bonne précision et une borne faible de l'erreur de reconstruction de l'enveloppe visuelle.

- Pour répondre au exigences d'un problème de modélisation et engendrer une surface fermée et sans auto-intersections, nous proposons deux algorithmes, l'approche hybride qui se base sur une triangulation de Delaunay de points sur la surface de l'enveloppe visuelle, et la méthode de reconstruction d'enveloppes visuelles polyédriques, qui calcule exactement l'intersection des cônes visuels associés à un jeu de vues, en calculant d'une part les segments de vue associés au sommets des contours dans les images, et d'autre part les arêtes d'intersection de cônes, de proche en proche. Cette dernière méthode est comparée à l'état de l'art, et évaluée en terme de rapidité et de fiabilité. Les approches proposées permettent de rivaliser en terme de fiabilité et de surpasser en rapidité les algorithmes existants, y compris ceux utilisés de CGAL qui utilisent l'arithmétique exacte pour calculer des intersections. En outre les méthodes proposées sont quasi temps réel.

- Nous constatons que l'un des problèmes difficiles souvent sous-estimé dans la littérature provient de la segmentation des images et l'extraction de silhouettes. Pour mieux intégrer l'information multi-vue et l'information temporelle de la séquence, et présenter une meilleure robustesse aux divers bruits dans les images, nous proposons une approche statistique basée sur les grilles d'occupation, qui permet une fusion de l'information et une modélisation complète de la chaîne d'incertitude dans le problème de modélisation de forme à partir de silhouettes.

- Nous proposons une réflexion sur l'architecture parallèle de systèmes de vision multi-vue, qui prend en compte le parallélisme au sein des tâches de vision. Des solutions sont données pour le problème de modélisation multi-vue dans le contexte d'une plate-forme multi-caméra et multi-PC, dans le cadre d'applications de réalité virtuelle. Les résultats présentés peuvent inspirer le traitement distribué d'autres tâches de vision.

Plusieurs perspectives nous semblent ouvertes par ce travail. Nous pensons par exemple qu'il est possible d'améliorer l'algorithme de reconstruction d'enveloppes visuelles polyédriques présenté à partir de silhouettes polygonales, pour obtenir une meilleure complexité asymptotique en nombre d'opérations requises.

D'autre part, l'approche statistique proposée pour la reconstruction de forme à partir de silhouettes soumises à l'incertitude admet des extensions possibles pour intégrer une information de cohérence temporelle que nous n'avons que partiellement abordées. De plus, le nouveau paradigme proposé au cours du chapitre sur cette méthode, qui est celui d'une soustraction de fond multi-vue, admet très certainement d'autres formulations intéressantes et utiles, qui méritent d'être ex-

plorées.

Enfin, nous avons exploré l'implémentation temps réel de telles approches à travers une parallélisation massive des tâches. D'autres algorithmes peuvent bénéficier de ce travail pour construire des systèmes puissants de traitement en vision. Mais l'implémentation temps réel peut aussi être effectuée sur GPU, dont les dernières générations ouvrent des perspectives de traitement nouvelles de parallélisation avec l'utilisation de plusieurs cartes.

Nous espérons donc que les résultats proposés seront utiles à la communauté scientifique. Nous sommes confiants dans le fait qu'ils ouvrent de nouvelles perspectives de recherche.

A.

Caméras à sténopé et géométrie projective

Dans toute cette thèse, nous nous plaçons dans le contexte où N caméras calibrées voient une scène comportant un ou plusieurs objets d'intérêt. Ces objets sont ceux dont on veut reconstruire la forme. Nous supposons que les caméras sont *à sténopé*, c'est à dire que leur système optique se résume à un point, appelé *centre optique*, ou centre de projection. Le modèle de la caméra à sténopé est illustré en figure A.1. Les images se forment dans un plan, dit *plan image* (ou *plan rétinien*), par projection des rayons passant par le centre optique. Chaque point de l'image formée correspond donc à une droite de projection, que l'on appelle communément *ligne de vue* de ce point. Comme expliqué dans les ouvrages de référence [50, 34], cette projection peut s'exprimer par une relation d'algèbre linéaire, dans un espace dit *projectif*. Tout point d'un espace Euclidien de $\mathbb{R}^d$ y est représenté par un jeu de $d+1$ coordonnées, dites *homogènes*. La dimension supplémentaire correspond à un facteur d'échelle : à un point de l'espace $\mathbb{R}^d$ correspond une famille de points équivalents dans l'espace projectif, au facteur d'échelle près. Si un point X possède pour coordonnées euclidiennes dans $\mathbb{R}^3$ le vecteur $(x\,y\,z)^\top$, alors on peut exprimer X en coordonnées homogènes, pour tout w non nul :

$$X \sim \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} \sim \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{A.1}$$

Le coefficient w est un facteur d'échelle, et $\sim$ symbolise l'*égalité projective*, à un facteur d'échelle près, propre à l'espace projectif.

En particulier, dans le contexte de notre problème nous manipulons principalement des entités de l'espace 3D et de l'espace image.

Nos vues sont calibrées, c'est à dire que nous avons connaissance de la position et des paramètres de la caméra. Nous savons donc exprimer la relation linéaire

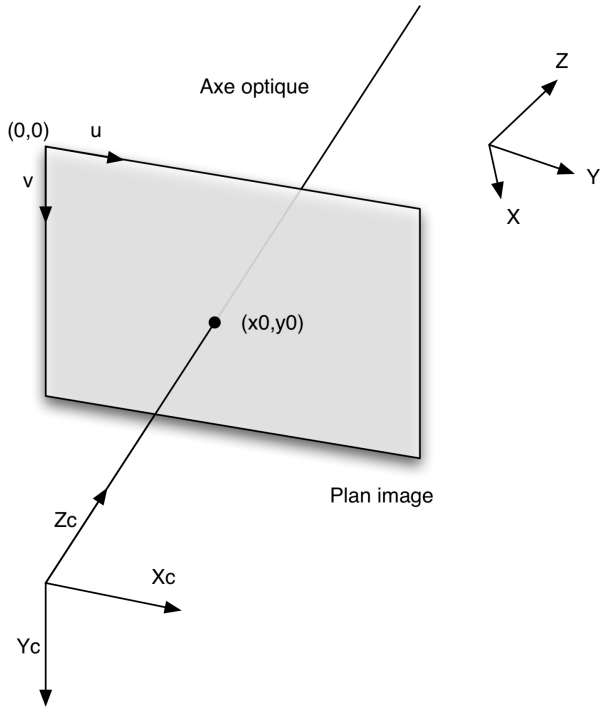


FIG. A.1 – Caméra à sténopé

entre les coordonnées homogènes d'un point X, et celles de sa projection x^i dans l'image i, sous la forme d'une matrice 3×4 que nous noterons P^i. Dans le cas de la caméra à sténopé, P^i est classiquement décomposée en deux matrices K^i et M^i [50, 34].

De manière générale nous prenons pour convention, tout au long de ce document, de noter le numéro des images en exposant quand cela a un sens.

K^i, matrice triangulaire supérieure de dimension 3×3, est la matrice de *paramètres intrinsèques*, et modélise l'état interne de la caméra, indépendant de sa position. Les paramétrages de cette matrice font intervenir, selon les modèles, de quatre à sept paramètres, qui permettent d'exprimer la distance focale de la caméra, l'échelle des pixels dans l'image, la position du point principal de la caméra (où passe l'axe optique), et éventuellement un coefficient permettant de rendre compte d'une déformation trapézoïdale de l'image formée (skew). La matrice M^i, de dimension 3×4, est la matrice des *paramètres extrinsèques*, qui permettent de représenter la pose de la caméra. Cette matrice possède un paramétrage à six degrés de liberté, dont trois en rotation et trois en translation. Elle peut ainsi elle-même être décomposée en deux termes, une matrice de rotation R^i ainsi qu'un vecteur de $\mathbb{R}^3$, noté $\mathbf{t}^i$, définissant la position du centre optique

dans l'espace. Cette matrice est en définitive une matrice de changement de repère (ou de déplacement rigide), permettant de passer du repère de la scène à celui de la caméra considérée.

Au final, la relation entre un point X et sa projection x^i dans le plan image de la caméra i peut s'écrire :

$$x^i \sim \mathsf{P}^i X \tag{A.2}$$

$$\sim \begin{pmatrix} \mathsf{K}^i & \mathbf{0} \end{pmatrix} \mathsf{M}^i X \tag{A.3}$$

$$\sim \begin{pmatrix} \mathsf{K}^i & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathsf{R}^i & -\mathsf{R}^i \mathbf{t}^i \\ \mathbf{0}^\top & 1 \end{pmatrix} X \tag{A.4}$$

Naturellement, ces modèles interviennent pour tous types de calculs en rapport avec l'enveloppe visuelle, tout au long de cette thèse.

Il est intéressant de savoir comment calculer une représentation de la ligne de vue associée à un point image, à partir de ce modèle. Un tel calcul intervient par exemple dans la construction des segments de vue. Toute ligne de vue passe par le centre optique de la caméra i associée, par définition. Il nous suffit donc de calculer la direction et le sens de cette ligne de vue, pour une détermination complète de celle-ci. Un vecteur $\mathbf{d}$ donnant la direction de la ligne de vue associée à un point image p peut être obtenu de la manière suivante [50, 34] :

$$\mathbf{d} = {\mathsf{R}^i}^\top {\mathsf{K}^i}^{-1} p \tag{A.5}$$

De plus, si l'on écrit p avec un facteur d'échelle w sous la forme $p = (wx\ wy\ w)^\top$, $\mathbf{d}$ donne également le sens de cette ligne de vue, qui pointe vers la scène si et seulement si $\mathsf{K}^i_{3,3} \cdot w > 0$. Ceci garantit que le vecteur $\mathbf{d}$ corresponde à un vecteur dont la troisième coordonnée est positive dans le repère de la caméra.

Bibliographie

[1] b-rep library. http ://breplibrary.sourceforge.net/.

[2] Cgal library. http ://www.cgal.org/.

[3] Gnu multi-precision library. http ://www.swox.com/gmp/.

[4] Gnu triangulated surface library. http ://gts.sourceforge.net/.

[5] Opengl utility library (glu). http ://www.opengl.org.

[6] Quick hull library. http ://www.qhull.org/.

[7] Twin library. http ://www.cadlab.ecn.purdue.edu/twin/.

[8] D. Arita and R.-I. Taniguchi. RPV-II : A Stream-Based Real-Time Parallel Vision System and Its Application to Real-Time Volume Reconstruction. 2001.

[9] D. Attali and J.-D. Boissonnat. A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. Rapport de recherche 4453, INRIA, 2002.

[10] N. Ayache. *Vision stéréoscopique et perception multisensorielle. Applications à la robotique mobile*. InterEditions, 1989.

[11] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, CS Dept, Stanford U., Oct. 1974. AIM-249, STAN-CS-74-463.

[12] B. G. Baumgart. A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, volume 44, pages 589–596, 1975.

[13] J. S. D. Bonet and P. A. Viola. Roxels : Responsibility weighted 3d volume reconstruction. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, (Greece)*, volume I, pages 418–425, Sept. 1999.

[14] E. Borovikov, A. Sussman, and L. Davis. A High Performance Multi-Perspective Vision Studio. In *17th Annual ACM International Conference on Supercomputing, San Francisco (USA)*, 2003.

[15] E. Boyer and M.-O. Berger. 3D surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3) :219–233, 1997.

[16] E. Boyer and J.-S. Franco. A Hybrid Approach for Computing Visual Hulls of Complex Objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, volume I, pages 695–701, 2003.

[17] M. Brady and A. Yuille. An Extremum Principle for Shape from Contour. *IEEE Transactions on PAMI*, 6(3) :288–301, 1984.

[18] M. Brand, K. Kang, and D. B. Cooper. Algebraic solution for the visual hull. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington DC, (USA)*, pages 30–35, 2004.

[19] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for the Space Carving algorithm. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, (Canada)*, volume I, pages 388–393, 2001.

[20] B. Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4) :671–696, 1992.

[21] G. Cheung, S. Baker, and T. Kanade. Visual Hull Alignment and Refinement Across Time : A 3D Reconstruction Algorithm Combining Shape-From-Silhouette with Stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, 2003.

[22] G. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, (USA)*, volume II, pages 714 – 720, June 2000.

[23] L. P. Chew. Constrained delaunay triangulations. In *SCG '87 : Proceedings of the third annual symposium on Computational geometry*, pages 215–222, New York, NY, USA, 1987. ACM Press.

[24] C. Chien and J. Aggarwal. Volume/surface octress for the representation of three-dimensional objects. *Computer Vision, Graphics and Image Processing*, 36(1) :100–113, 1986.

[25] P. Cignoni, C. Montani, R. Perego, and R. Scopigno. Parallel 3D Delaunay Triangulation. *Computer Graphics Forum*, 12(3) :129–142, 1993.

[26] R. Cipolla and A. Blake. Surface Shape from the Deformation of Apparent Contours. *International Journal of Computer Vision*, 9 :83–112, 1992.

[27] C. Coue. *Modèle bayésien pour l'analyse multimodale d'environnements dynamiques et encombrés : application à l'assistance à la consuite automobile en milieu urbain.* PhD thesis, Institut National Polytechnique de Grenoble, Dec. 2003.

[28] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. In *Proceedings of IEEE*

Conference on Computer Vision and Pattern Recognition, Santa Barbara, (USA), page 601, Washington, DC, USA, 1998. IEEE Computer Society.

[29] I. Debled-Rennesson and J. Reveillès. A linear algorithm for segmentation of digital curves. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(4) :635–662, 1995.

[30] I. Debled-Rennesson, S. Tabbone, and L. Wendling. Fast Polygonal Approximation of Digital Curves. volume I, pages 465–468, 2004.

[31] C. Dyer. Volumetric Scene Reconstruction from Multiple Views. In L. Davis, editor, *Foundations of Image Understanding*, pages 469–489. Kluwer, Boston, 2001.

[32] H. Edelsbrunner. *Geometry and topology for mesh generation.* Cambridge University Press, New York, NY, USA, 2001.

[33] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer, Special Issue on Autonomous Intelligent Machines*, 22(6) :46–57, June 1989.

[34] O. Faugeras. *Three-Dimensional Computer Vision : A Geometric Viewpoint.* Artificial Intelligence. MIT Press, Cambridge, 1993.

[35] S. Fortune. Polyhedral modelling with exact arithmetic. In *SMA '95 : Proceedings of the third ACM symposium on Solid modeling and applications*, pages 225–234, New York, NY, USA, 1995. ACM Press.

[36] S. Fortune and C. V. Wyk. Efficient exact arithmetic for computational geometry, 1993.

[37] A. François and G. Médioni. A Modular Software Architecture for Real Time Video Processing. pages 35–49, 2001.

[38] J.-S. Franco and E. Boyer. Une approche hybride pour calculer l'enveloppe visuelle d'objets complexes. In *Actes du 9-ème congrès francophone ORASIS'03, Gerardmer (France).*

[39] J.-S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In *Proceedings of the British Machine Vision Conference, Norwich (UK)*, pages 329–338, Sept. 2003.

[40] J.-S. Franco and E. Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, (China)*, volume 2, pages 1747–1753, oct 2005.

[41] J.-S. Franco and E. Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. Technical Report 5551, INRIA, April 2005.

[42] J.-S. Franco, C. Ménier, E. Boyer, and B. Raffin. A distributed approach for real-time 3d modeling. *CVPR Workshop on Real-Time 3D Sensors and their Applications*, 2004.

[43] N. Friedman and S. Russell. Image Segmentation in Video Sequences : A Probabilistic Approach. In *Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence*, 1997.

[44] P. Giblin and R. Weiss. Reconstruction of Surfaces from Profiles. In *Proceedings of the First International Conference on Computer Vision, London*, pages 136–144, 1987.

[45] B. Goldlücke and M. Magnor. Joint 3-d reconstruction and background separation in multiple views using graph cuts. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, volume I, pages 683–694, June 2003.

[46] B. Goldlücke and M. A. Magnor. Space-time isosurface evolution for temporally coherent 3d reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington DC, (USA)*.

[47] K. Grauman, G. Shakhnarovich, and T. Darrell. A bayesian approach to image-based visual hull reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, volume I, pages 187–194, June 2003.

[48] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI : Portable Parallel Programming with the Message-Passing Interface*. Scientific and Engeneering Computation Series. The MIT Press, 1994.

[49] P. Hachenberger and L. Kettner. Boolean operations on 3d selective nef complexes : optimized implementation and experiments. In *SPM '05 : Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 163–174, New York, NY, USA, 2005. ACM Press.

[50] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000.

[51] C. M. Hoffmann, J. E. Hopcroft, and M. E. Karasick. Robust set operations on polyhedral solids. *IEEE Computer Graphics and Applications*, 9(6) :50–59, 1989.

[52] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *ACM Computer Graphics (Proceedings SIGGRAPH)*, volume 26(2), pages 71–78, July 1992.

[53] T. Horprasert, D. Harwood, and L. Davis. A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection . In *IEEE ICCV'99 FRAME-RATE WORKSHOP*, 1999.

[54] Intel. Open Source Computer Vision Library. http ://www.intel.com/research/mrl/research/opencv/.

[55] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *MOTION*

'02 : Proceedings of the Workshop on Motion and Video Computing, page 22, Washington, DC, USA, 2002. IEEE Computer Society.

[56] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic engineering*, 35, Mar 1960.

[57] Y. Kameda, T. Taoda, and M. Minoh. High Speed 3D Reconstruction by Spatio Temporal Division of Video Image Processing. *IEICE Transactions on Informations and Systems*, (7) :1422–1428, 2000.

[58] K. Kanatani and T. Chou. Shape from texture : General principal. *Artificial Intelligence*, 38 :1–48, 1989.

[59] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active Contour Models. *International Journal of Computer Vision*, 1 :321–331, 1988.

[60] R. Keriven. A variational framework for shape from contour, research report 2002-221b. Technical report, CERMICS, June 2002.

[61] J. Koenderink. What Does the Occluding Contour Tell us About Solid Shape ? *Perception*, 13 :321–330, 1984.

[62] K. Kutulakos and S. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3) :199–218, 2000.

[63] D. H. Laidlaw, W. B. Trumbore, and J. F. Hughes. Constructive solid geometry for polyhedral objects. In *SIGGRAPH '86 : Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 161–170, New York, NY, USA, 1986. ACM Press.

[64] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2) :150–162, Feb. 1994.

[65] S. Laveau and O. Faugeras. Oriented projective geometry for computer vision. In *Proceedings of Fourth European Conference on Computer Vision, Cambridge, (England)*, 1996.

[66] S. Lazebnik, E. Boyer, and J. Ponce. On How to Compute Exact Visual Hulls of Object Bounded by Smooth Surfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Kauai, (USA)*, volume I, pages 156–161, December 2001.

[67] M. Li, M. Magnor, and H.-P. Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Proceedings of Graphics Interface'2003*, Halifax, Canada, 2003.

[68] M. Li, M. Magnor, and H.-P. Seidel. Improved hardware-accelerated visual hull rendering. In *Proceedings of the Vision, Modeling, and Visualization Conference (VMV 2003), München (Germany)*, 2003.

[69] M. Li, M. Magnor, and H.-P. Seidel. A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls. In *GI '04 : Proceedings of*

the 2004 conference on Graphics interface, pages 41–48, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[70] C. Liang and K.-Y. K. Wong. Complex 3d shape recovery using a dual-space approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Diego, (USA)*, pages 878–884, 2005.

[71] D. Margaritis and S. Thrun. Learning to locate an object in 3d space from a sequence of camera images. In *International Conference on Machine Learning*, pages 332–340, 1998.

[72] W. Martin and J. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on PAMI*, 5(2) :150–158, 1983.

[73] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. In *Eurographics Workshop on Rendering*, 2001.

[74] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image Based Visual Hulls. In *ACM Computer Graphics (Proceedings Siggraph)*, pages 369–374, 2000.

[75] W. Matusik, L. McMillan, and S. Gortler. An Efficient Visual Hull Computation Algorithm. Technical report, MIT LCS, Feb. 2002.

[76] C. Ménier, J. Allard, J.-S. Franco, B. Raffin, and E. Boyer. Marker-less real time 3d modeling for virtual reality. *Immersive Projection Technology*, 2004.

[77] C. Ménier, J.-S. Franco, E. Boyer, and B. Raffin. Modélisation tridimensionnelle temps-réel et distribuée. In *Actes des Journées ORASIS*, May 2005.

[78] H. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 1996.

[79] C. Ménier, J.-S. Franco, B. Raffin, and E. Boyer. The GrImage Platform : A Mixed Reality Environment for Interactions. In *Proceedings of International Conference on Computer Vision Systems (ICVS'06), New York, January 2006*, Jan. 2006.

[80] P. Narayanan, P. Rander, and T. Kanade. Constructing Virtual Wolrds Using Dense Stereo. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, (India)*, pages 3–10, 1998.

[81] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. *SIGGRAPH Computer Graphics*, 24(4) :115–124, 1990.

[82] W. Nef. *Beiträge zur Theorie der Polyeder*. Herbert Lang, Bern, 1978.

[83] W. Niem. Automatic Modeling of 3D Natural Objects from Multiple Views. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, Germany*, 1994.

[84] A. P. Pentland. Linear Shape from Shading. *International Journal of Computer Vision*, 4 :153–162, 1990.

[85] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40(1) :1–29, 1987.

[86] A. A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling : Boundary evaluation and merging algorithms. *Proc. IEEE*, 73(1) :30–44, Jan. 1985.

[87] C. Rother, V. Kolmogorov, and A. Blake. "grabcut" : interactive foreground extraction using iterated graph cuts. *ACM Transaction on Graphics (Special issue : SIGGRAPH'04)*, 23(3) :309–314, 2004.

[88] S. Rowe and A. Blake. Statistical mosaics for tracking. *Image and Vision Computing*, 14 :549–564, 1996.

[89] R. Seidel. The nature and meaning of perturbations in geometric computing. In *STACS '94 : Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 3–17, London, UK, 1994. Springer-Verlag.

[90] S. Seitz and C. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Juan, (Puerto Rico)*, pages 1067–1073, 1997.

[91] J. Sérot and D. Ginhac. Skeletons for parallel image processing : an overview of the skipper project. *Parallel Computing*, 28(12) :1685–1708, 2002.

[92] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. In *Discrete and Computational Geometry*, volume 18, pages 305–363, 1997.

[93] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafe. A Survey of Methods for Volumetric Scene Reconstruction from Photographs. In *International Workshop on Volume Graphics*, 2001.

[94] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, (USA)*, pages 345–353, 2000.

[95] S. K. Srivastava. Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics and Image Processing*, 49(1) :68–84, 1990.

[96] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of IEEE Conference on Computer Vision*

and Pattern Recognition, Fort Collins, (USA), volume II, pages 246–252, June 1999.

[97] S. Sullivan and J. Ponce. Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs using Triangular Splines. *IEEE Transactions on PAMI*, pages 1091–1096, 1998.

[98] R. Szeliski. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 58(1) :23–32, 1993.

[99] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower : Principles and Practice of Background Maintenance. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, (Greece)*, pages 255–261, 1999.

[100] F. Ulupinar and R. Nevatia. Perception of 3D Surfaces from 2D Contours. *IEEE Transactions on PAMI*, 15(1) :3–18, 1993.

[101] R. Vaillant and O. Faugeras. Using Extremal Boundaries for 3-D Object Modeling. *IEEE Transactions on PAMI*, 14(2) :157–173, Feb. 1992.

[102] L. G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8) :103–111, August 1990.

[103] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6d. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, (USA)*, June 2000.

[104] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder : Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7) :780–785, 1997.

[105] Q.-Z. Wu and B.-S. Jeng. Background subtraction based on logarithmic intensities. *Pattern Recognition Letters*, 23(13) :1529–1536, 2002.

[106] C.-K. Yap. Towards exact geometric computation. *Computational Geometry, Theory and Applications*, 7(1-2) :3–23, 1997.

[107] A. Yilmaz, X. Li, and M. Shah. Object contour tracking using level sets. In *Proceedings of the 6th Asian Conference on Computer Vision, Jeju Island, (Korea)*, Jan. 2004.

[108] G. Zeng and L. Quan. Silhouette extraction from multiple images of an unknown background. In *Proceedings of the 6th Asian Conference on Computer Vision, Jeju Island, (Korea)*, Jan. 2004.