Modélisation de scènes dynamiques à partir de plusieurs caméras

Edmond Boyer

To cite this version:

Edmond Boyer. Modélisation de scènes dynamiques à partir de plusieurs caméras. Interface homme-machine [cs.HC]. Université Joseph-Fourier - Grenoble I, 2007. tel-00584185

HAL Id: tel-00584185
https://theses.hal.science/tel-00584185

Submitted on 7 Apr 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ JOSEPH FOURIER de GRENOBLE

HABILITATION À DIRIGER DES RECHERCHES

Spécialité : "Informatique"

présentée par

Edmond Boyer

Modélisation de scènes dynamiques à partir de plusieurs caméras

soutenue publiquement le 19 janvier 2007 devant le jury composé de :

M.	**Roger Mohr**	INP Grenoble
M.	**Michel Dhome**	CNRS Clermont-Ferrand
M.	**Francis Schmitt**	ENST Paris
M.	**Marie-Odile Berger**	INRIA Nancy
M.	**Roberto Cipolla**	Université de Cambridge
M.	**Jean Ponce**	ENS Paris

Table des matières

Chapitre 1

Introduction

Ce document présente des travaux de recherche réalisés durant la période allant de 2000 à 2005 et portant sur la modélisation de scènes dynamiques à partir de plusieurs caméras. Ce thème constitue un axe de recherche actif du domaine de la vision par ordinateur depuis de nombreuses années. L'objectif général est la capacité à *produire une description opératoire du contenu d'une scène à partir de plusieurs vidéos.* Au même titre que dans la vision humaine, et selon une théorie computationnelle de celle-ci, cette description a pour objet d'alimenter des processus cognitifs qui mènent de la perception à l'action. Le contenu de cette description peut alors être varié, en fonction de l'action visée. Il peut s'agir d'information sur les formes, comme la géométrie et l'apparence, pour la navigation d'un robot ou pour *virtualiser* la réalité et en modifier les aspects visuels. La description peut aussi porter sur la sémantique de la scène, par exemple ce qui est en mouvement et de quelle manière, pour en permettre son interprétation.

Motivation
Les applications de la modélisation de scènes dynamiques sont multiples que ce soit en video-surveillance, en robotique ou, plus proche de nos préoccupations, dans le domaine du multimédia. C'est en effet dans ce dernier que réside la motivation principale des travaux présentés ici, à savoir la mise en oeuvre d'applications interactives où univers réels et virtuels se mélangent. Ces applications nécessitent une description de la réalité qui soit compatible avec celle des environnements virtuels ; et du niveau de compatibilité qui existe entre ces représentations dépend le niveau d'interaction accessible. Typiquement une description des formes observées permet le mélange de formes réelles et virtuelles et la visualisation interactive du résultat, mais limite les interactions possibles à des contacts simplifiés entre objets réels et virtuels. Pour des interactions plus avancées, comme la manipulation d'objets virtuels, la description devra être plus riche et contenir, par exemple, des informations sur le mouvement de la main. À un niveau encore supérieur, une description des activités, ou des actions, qui se déroulent dans la scène observée permettra à un système de détecter une situation et d'y réagir de manière automatique.

Problématique
La problématique qui a été la notre dans ces travaux concerne l'acquisition

d'information pour des applications interactives. Cette acquisition peut être vue comme une chaîne de traitement de l'information dont les entrées sont les images produites au cours du temps par plusieurs caméras, éventuellement en nombre important, et dont la sortie est une séquence temporelle de modèles de types variables : des surfaces ou des modèles articulés typiquement. Les difficultés qui se présentent dans ce cadre sont multiples. En effet, si la géométrie de points observés par plusieurs caméras est bien connue, plusieurs ouvrages traitent de ce problème[Har 00, Fau 01, For 03], le problème de la modélisation de surfaces non rigides dans le temps reste ouvert, d'autant plus si l'on considère la contrainte temps-réel souvent nécessaire aux systèmes interactifs. Dans les très nombreux travaux qui couvrent ce thème, assez peu considèrent le problème pratique de l'acquisition et du traitement de vidéo multiples, encore moins proposent des solutions qui fonctionnent en temps réel. Néanmoins, nos travaux s'inscrivent dans la lignée de ceux, précurseurs, de l'université de Carnegie Mellon sur la *virtualisation*[Nar 98, Che 00] d'évènements réels, et de ceux de l'ETH de Zurich autour du projet *Blue C* [Gro 03, WÖ4] qui modélise et immerge dans un espace virtuel des personnes distantes à l'aide de caméras et d'écrans LCD géants. Dans les processus hors-lignes, citons l'approche du Max Planck Institute sur *Free Viewpoint Video*[Car 03b] qui consiste à construire un modèle articulé et texturé d'une personne à partir de plusieurs vidéos. Enfin toujours dans les processus hors-lignes, l'université de Surrey [Sta 03, Hil 04] s'est aussi intéressée à la modélisation d'avatars à partir de vidéos de personnes.

Contributions

Dans les travaux présentés dans ce document, nous considérerons le problème de la modélisation à partir de plusieurs images de manière dynamique, c'est à dire pour des séquences d'images dans le temps. Notre ambition dans ce domaine est double : il s'agit de proposer un cadre technologique qui permette la réalisation d'applications aussi bien temps réel que hors-ligne ; il s'agit par ailleurs de proposer des solutions aux problèmes scientifiques qui se posent et de mettre en oeuvre ces solutions. Nos contributions sur ce thème sont multiples et concernent les aspects scientifiques : que ce soit sur la reconstruction à partir de silhouettes ou sur la capture de mouvement, ainsi que les aspects pratiques comme la mise en oeuvre de la plateforme d'acquisition Grimage. D'une manière plus générale, et en rapport avec les travaux mentionnés précédemment, nos contributions ont mené à la réalisation d'une plateforme d'interactions tridimensionnelles où l'ensemble du corps agit sur les mondes virtuels.

Approche

Les problèmes que pose la mise en oeuvre de ces applications sont nombreux. Ils vont de l'acquisition d'images à la modélisation tridimensionnelle ainsi qu'à la gestion des interactions. Ce document traite plus particulièrement des problèmes scientifiques concernant la modélisation. S'il n'y pas de philosophie générale dans les travaux que nous avons menés sur ce thème, quelques directions fortes se dégagent néanmoins. En premier lieu notre approche a une structure ascendante (*bottom-up*) où les données, les images, servent à produire des informations tridimensionnelles sur les formes observées en faisant intervenir peu de connaissances *a priori*. L'idée ici est de produire une description géométrique quelque soit le contenu de la scène. Ce modèle est bien adapté au contexte qui est le notre, à savoir des scènes dynamiques et donc des contenus tridimensionnels pour lesquels il est difficile d'avoir une connaissance *a priori*. En deuxième lieu, notre approche modélise les formes observées au sens de surfaces en mouvement dans l'espace et non de points ou de segments. Pour cela, les informations géométriques que nous considérons dans les images sont les silhouettes qui caractérisent la projection de l'ensemble de la forme observée et non d'une partie de cette dernière.

Contexte matériel

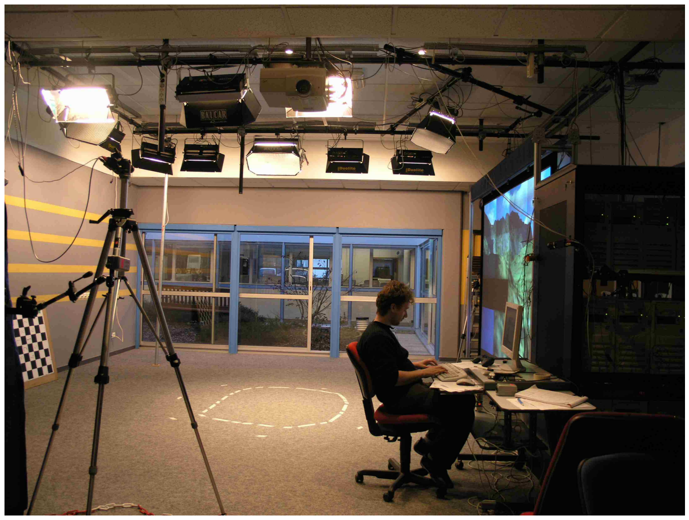


FIG. 1.1 – La plateforme Grimage.

Les recherches présentés dans ce document sont très fortement liées à la plateforme d'acquisition Grimage (figure 1.1). Cette plateforme est constituée de PCs en grappe, de caméras standard et de multiples projecteurs constituant un mur d'écrans de haute résolution. L'ensemble des tâches à gérer sont distribuées sur l'ensemble des ressources disponibles à l'aide d'un intergiciel[All 05]. Cette plateforme a été mise en oeuvre durant la période des travaux reportés ici et a permis d'acquérir les données nécessaires à la validation de nos méthodes. Plus remarquablement, elle a montré, au travers de nombreuses démonstrations publiques, l'impact qu'une application interactive même simple pouvait avoir sur les gens et a confirmé l'intérêt que présente ses applications pour tout un chacun.

Contenu du document

L'architecture générale de la chaîne d'acquisition que nous avons constituée est montrée figure 1.2. Les différents éléments de cette chaîne ont fait l'objet de travaux de recherche et développement. Ce document traite uniquement des contributions scientifiques qui ont été faites pour la mise en oeuvre de cette chaîne d'acquisition. Ces contributions ne concernent pas toutes les parties de la chaîne d'acquisition mais une majorité d'entre elles. En particulier l'acquisition

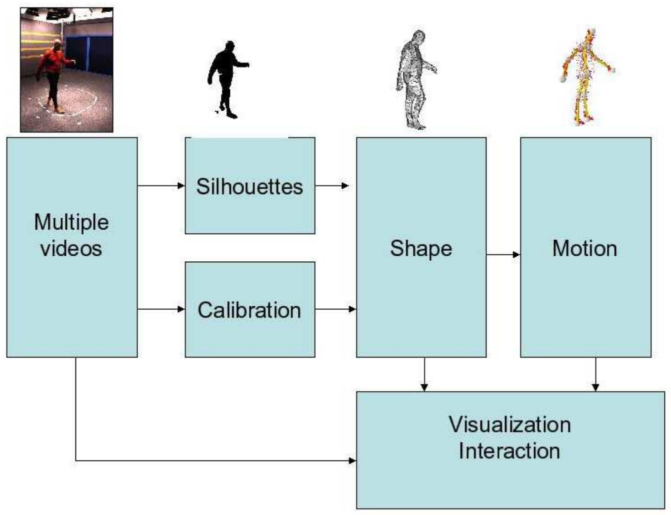


FIG. 1.2 – L'architecture du système d'acquisition.

d'images, l'extraction de silhouettes ou le calibrage classique des caméras pour lesquels des solutions satisfaisantes existent ne seront pas évoqués ici, même si elles ont fait l'objet d'un travail de développement important. Le document est constitué d'une partie introductive en Français suivi d'articles publiés dans la période de 2000 à 2005. L'organisation des diverses parties est celle de la chaîne d'acquisition. Dans un premier temps, les notions utiles à la compréhension du document seront introduites dans le chapitre 1. Le problème du calibrage de caméra à l'aide des silhouettes, qui constitue une excellente alternative à une calibration standard, sera ensuite évoquée dans le chapitre 2. L'estimation des formes observées qui constitue une partie importante de nos travaux sera traitée en profondeur dans le chapitre 3. L'estimation du mouvement lorsqu'une personne est observée sera enfin étudiée dans le chapitre 4 avant de livrer un bilan et des perspectives à ces travaux.

Chapitre 2

Définitions

Dans ce chapitre, nous introduisons diverses notions relatives aux silhouettes et utiles à la lecture de l'ensemble du document. Ces notions concernent les formes 3D associées à un ensemble de silhouettes, ainsi que l'étude de leurs structures topologiques. En dehors de Laurentini [Lau 94] qui a donné la définition théorique de l'enveloppe visuelle dans le cas où une infinité de point de vue existent, peu d'auteurs se sont intéressés aux formes dans l'espace définis par un ensemble de silhouettes. C'est l'objet des travaux introduits dans ce chapitre. Il s'agit de fournir non seulement un cadre formel à l'étude de l'enveloppe visuelle mais aussi de mettre en évidence sa structure et d'introduire les formes visuelles dont l'enveloppe visuelle est la boite englobante. Les définitions correspondantes sont le résultat d'études réalisées à l'INRIA ainsi qu'en collaboration avec Svetlana Lazebnik et Jean Ponce de l'université de l'Illinois à Urbana Champaign. Elles se trouvent dans plusieurs publications [Laz 01, Boy 03, Lap 06] qui se trouvent respectivement pages [41,48,56] de ce document.

2.1 Les primitives

Nous considérons un ensemble d'images calibrées dans lesquelles les zones d'intérêts - correspondant à un ou plusieurs objets de la scène observée - ont été identifiées. Les primitives qui nous intéressent sont celles qui permettent de décrire les régions d'intérêt dans l'image, ainsi que les régions correspondantes dans l'espace, c'est à dire celles se projetant dans une ou plusieurs silhouettes.

2.1.1 Les primitives associées à un point de vue

La région 2D dans laquelle un objet, un tore par exemple dans la figure 2.1, se projette dans une image constitue une silhouette de cet objet. Chaque silhouette définie une région 3D, son cône de vue, qui contient l'ensemble des points de l'espace se projetant sur la silhouette, comme cela est illustré dans la partie droite de la figure 2.1. Pour une scène complexe, la silhouette peut être constituée de plusieurs parties présentant des trous et non nécessairement connexes dans l'image. Pour gérer ces situations complexes, nous associons à chaque contour fermé de l'image un cône et nous considérons le cône de vue d'une silhouette comme l'intersection des cônes des contours constituant cette silhouette (voir [Boy 03] page 48). L'intérêt de cette démarche est de pouvoir facilement ramener le calcul de l'intersection de plusieurs cônes de vue à celui de l'intersection de contours dans les images. Par ailleurs, un cône de vue est tangent à la surface de la scène observée le long d'une courbe appelée *contour d'occultation* ou *rim* en Anglais, comme illustré 2.2.

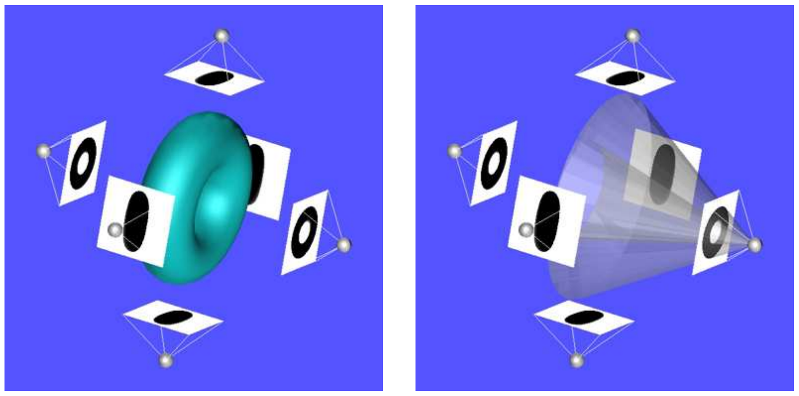


FIG. 2.1 – Les silhouettes, en noir, d'un tore et, à droite, le cône de vue associée à une silhouette.

2.1.2 Les primitives associées à plusieurs points de vue

Les surfaces de deux cônes de vues d'une même scène s'intersectent selon des courbes dites *courbes d'intersection de cônes.* Ces courbes d'intersection n'appartiennent pas à la surface observée, sauf aux *points frontières* [Cip 95] qui sont des points particuliers de la surface observée où 2, ou plus, contours d'occultations s'intersectent. Courbes d'intersection et contours d'occultations se rejoignent donc aux points frontières comme cela est illustré dans la figure 2.2. Les *bandes de cônes* ou *cône strips* sont les parties de la surface d'un cône de vue qui se trouvent à l'intérieur d'autres cônes.

Les surfaces de 3 cônes de vue s'intersectent en des points particuliers appelés *points triples* (voir l'exemple de la figure 2.3). Ces points n'appartiennent pas à la surface observée sauf dans le cas particulier où les points de vue des cônes et le point observé sont coplanaires. Dans ce cas, le point triple est aussi le point frontière des cônes considérés 2 à 2. Des situations de ce genre se présentent notamment lorsque les points de vue sont colinéaires. Les surfaces de 4 cônes de vue ne s'intersectent pas dans une situation générique ; le cas de points de vue colinéaires constituant un exemple de situation non-générique.

2.2 Les maillages

Plusieurs images, issues de différents points de vue, définissent plusieurs contours d'occultations sur la surface de la scène observée. Ces contours constituent un maillage que nous appelons simplement le *maillage des contours d'occultations*, ou le *rim mesh* en Anglais. Par ailleurs, les cônes de vue associés aux différents points de vue ont un volume commun dans l'espace, l'enveloppe

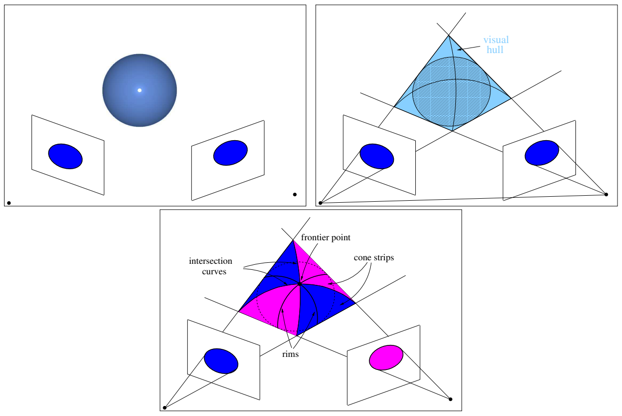


Fig. 2.2 – L'intersection de 2 cônes de vue : l'exemple d'un objet sphérique.

visuelle, délimité par un autre maillage constitué de courbes d'intersection de cônes, de points frontières et de points triples. Ces 2 maillages, l'un sur la surface observée et l'autre sur la surface de l'enveloppe visuelle, sont étroitement liés ; ils se rejoignent aux points frontière. Nous avons identifié ces maillages dans [Laz 01], page 41, ils permettent de comprendre la structure de l'enveloppe visuelle, et donc d'en faciliter son calcul. Ils permettent aussi de relier l'enveloppe visuelle à la surface observée, menant ainsi à la définition des formes visuelles.

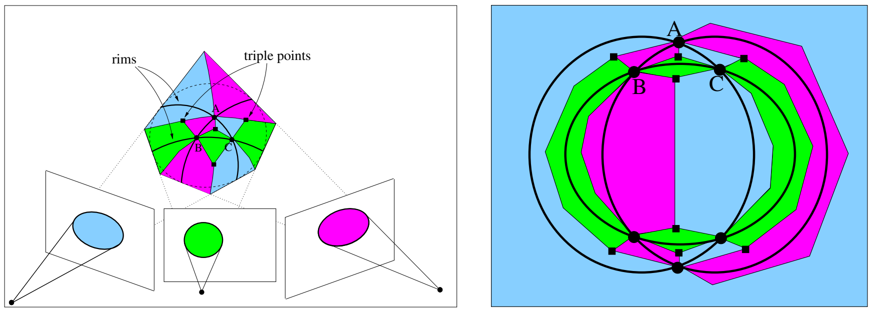


Fig. 2.3 – A gauche l'intersection de 3 cônes de vue pour un objet sphérique et à droite le graphe correspondant pour le maillage des contours (traits épais) et celui l'enveloppe visuelle (traits fins). Les points frontières sont représentés par des disques et les points triples par des carrés.

La figure 2.3 montre un exemple de ces maillages dans le cas d'un objet sphérique. le maillage des contours d'occultation relie entre eux les points frontières et le maillage de l'enveloppe visuelle relie points frontières et points triples au travers des courbes d'intersection de cônes. Sur la figure apparaissent aussi en couleurs les bandes de contribution des différents cônes. L'enveloppe visuelle est donc constituée de bandes de cônes à l'intérieur desquelles circulent les contours d'occultations, qui matérialisent le contact entre l'enveloppe visuelle et la surface. Cet exemple intuitif illustre un cas théorique simple. Dans la pratique, en raison des occultations, l'enveloppe visuelle est constituée de bandes de cônes qui ne contiennent pas nécessairement un contour d'occultation. Par ailleurs, toujours en pratique et en raison des imprécisions numériques, les cônes de vue ne sont pas exactement tangents et les points frontières deviennent alors des facettes connectant entre eux des points triples. De fait, comme nous le verrons dans la partie estimation 4, l'enveloppe visuelle *réelle* est constituée de facettes reliant des points triples.

2.3 Les formes dans l'espace

Nous nous intéressons dans cette partie aux formes 3D qui correspondent à un ensemble de silhouettes donné. Ces formes incluent bien sur l'enveloppe visuelle, qui est souvent définie de manière intuitive comme le volume maximal compatible avec les silhouettes, mais aussi d'autres formes dont celle observée qui se trouve à l'intérieur de l'enveloppe visuelle. L'enveloppe visuelle est en effet la boite englobante de toutes les formes compatibles avec un ensemble de silhouettes. Son intérêt est de fournir un modèle relativement proche de la scène observée lorsque le nombre de caméras est important. Néanmoins, il existe de meilleures approximations de la surface observée, dont les surfaces auxquelles l'enveloppe visuelle est tangente. L'objectif des travaux introduits dans cette partie est donc, dans un premier temps, de fournir une définition mathématique reliant les primitives définies précédemment aux enveloppes visuelles, ce qui en permet le calcul simplifié et rapide. Cette définition se trouve dans [Boy 03], page 48. Dans un deuxième temps, nous nous intéresserons aux formes visuelles, une famille plus large de formes dans l'espace qui inclut l'enveloppe visuelle et les formes auxquelles l'enveloppe visuelle est tangente. L'intérêt est de fournir une meilleure approximation de la surface observée, en particulier lorsque le nombre de caméras est faible, tout en conservant la rapidité de calcul. Les formes visuelles ont été étudiées dans [Lap 06], page 56. Cette partie résume les définitions, des détails se trouvent dans les publications [Boy 03, Lap 06]. Les méthodes d'estimation des formes sont quant à elles discutées dans le chapitre 4.

2.3.1 L'enveloppe visuelle

Le terme d'enveloppe visuelle, *visual hull* a été introduit par Laurentini [Lau 94] pour décrire l'objet maximal dont les silhouettes correspondent à celles données. Dans le cas où un objet est observé suivant une infinité de points de vue, l'enveloppe visuelle correspond alors à l'objet observé sans ses *concavités*[1]. Les contours d'occultations *glissent* en effet sur les concavités et les silhouettes ne fournissent donc aucune information sur ces parties. Une autre manière de définir l'enveloppe visuelle est de considérer le volume à l'intersection de l'ensemble des cônes de vue. Cette définition, équivalente à celle de l'objet maximal, s'avère en revanche plus pratique dans le cadre de l'estimation de l'enveloppe visuelle. C'est celle qui est utilisée par la majorité des méthodes d'estimation, en particulier les méthodes volumiques. En revanche, les méthodes précises déterminent la surface de l'enveloppe visuelle et considèrent les contours des silhouettes dans les images. Nous avons donc proposé dans [Boy 03] une définition de l'enveloppe

[1]Il est entendu ici les régions de l'espace dont les points ne peuvent être vus depuis des lignes de vue qui n'intersectent pas l'objet observé.

visuelle basée sur les contours dans les images. Cette définition étend par ailleurs le concept de l'enveloppe visuelle définie auparavant pour des caméras ayant un espace d'observation commun à celui de caméras ayant des points de vue quelconques. En effet, une caméra ne voit pas nécessairement l'ensemble de la scène et son influence dans le calcul de l'intersection des cônes de vue doit être limitée à son domaine de visibilité. Les détails se trouvent dans le papier, page 48.

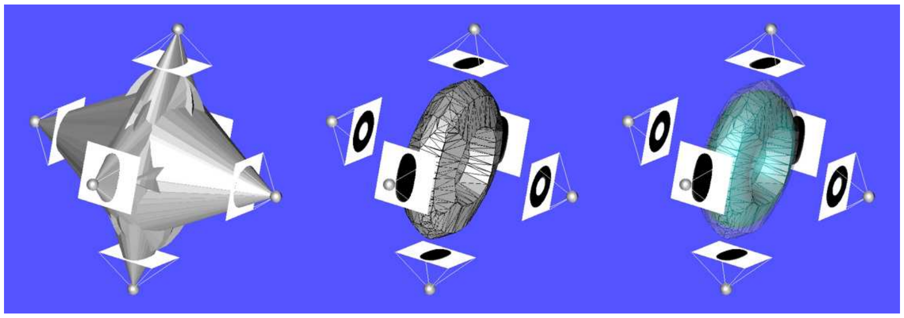


FIG. 2.4 – L'exemple du tore, de gauche à droite : les cônes de vues ; l'enveloppe visuelle intersection ; la surface du tore à l'intérieur de l'enveloppe visuelle.

2.3.2 Les formes visuelles

L'enveloppe visuelle constitue une boite englobante dont les faces sont tangentes à la surface observée. Comme cela a été dit précédemment, cette boite englobante est plus ou moins précise en fonction du nombre de point de vue. Des méthodes existent pour améliorer l'enveloppe visuelle à l'aide de l'information photo-métrique, lorsque celle-ci est cohérente dans plusieurs images, par exemple [Her 04, Sin 05, Fur 06]. Notre objectif ici est différent, il s'agit de fournir une meilleure approximation à partir des silhouettes uniquement. L'intérêt est que le modèle produit constitue une meilleure initialisation que l'enveloppe visuelle sans nécessiter l'usage d'informations parfois difficile à obtenir, comme par exemple la cohérence photo-métrique lorsque peu de vues sont disponibles. L'idée directrice est que l'enveloppe visuelle est constituée de bandes de cônes de vue et qu'à l'intérieur de ces bandes se situe le contour d'occultation où s'effectue le contact entre la surface et l'enveloppe visuelle. Les formes visuelles, *visual shapes*, partagent donc avec l'enveloppe visuelle la notion de bandes de cônes ; toutes les formes visuelles associées à ensemble donné de silhouettes possèdent les mêmes bandes de cônes mais de largeurs différentes. En particulier, en réduisant la largeur des bandes, on obtient, à la limite, des formes visuelles dont le contact avec l'enveloppe visuelle se réduit à une courbe à l'intérieur des bandes. Le choix de la position de la courbe à l'intérieur des bandes peut se faire ensuite en supposant la surface de la forme lisse par exemple.

Les formes visuelles englobent l'enveloppe visuelle qui matérialise la limite de ces formes pour un ensemble donné de silhouettes. Il est à noter que le processus de rétrécissement des bandes de cônes s'obtient naturellement, dans les parties non concaves, en augmentant le nombre de points de vue considéré lors du calcul de l'enveloppe visuelle. Une conséquence de cela est que pour un grand nombre de points de vue, la différence entre les formes visuelles sera faible. Plus de détails concernant les formes visuelles sont donnés dans [Lap 06] page56.

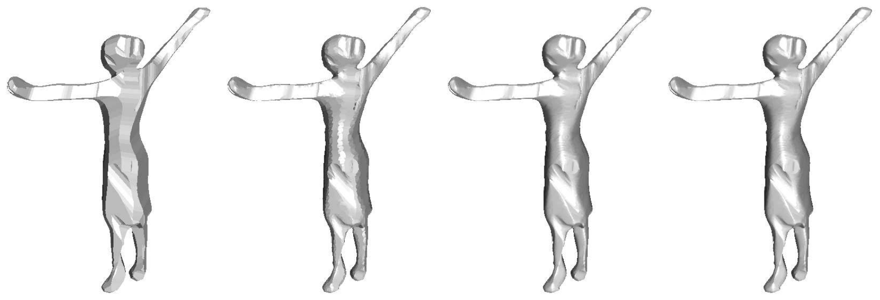


Fig. 2.5 – Exemples de formes visuelles, de gauche à droite : l'enveloppe visuelle et différent niveaux de subdivision de cette dernière. Toutes ces formes produisent les même silhouettes.

Chapitre 3

Estimation du calibrage à l'aide des silhouettes

Dans le chapitre précédent, nous avons introduit plusieurs notions liées aux silhouettes, en particulier les cônes de vue. Dans ce chapitre, nous allons voir que les cônes de vues d'une même scène vérifient certaines contraintes et que ces contraintes peuvent servir à déterminer le calibrage des caméras. Le calibrage est une étape importante des processus de modélisation à partir d'images. Il consiste à déterminer les caractéristiques et les poses des caméras impliquées dans le processus de modélisation. Ce calibrage peut être faible ; la structure projective de l'enveloppe visuelle peut, en effet, être déterminée à partir de la géométrie épipolaire orientée [Laz 01]. Néanmoins, dans un contexte pratique un calibrage complet est souvent nécessaire, en particulier pour les applications de réalité mixte où les modèles produits sont insérés dans des environnements virtuels. Le calibrage des caméras peut être obtenu par des méthodes standards, à l'aide d'objets de références dans la scène par exemple. La plateforme Grimage dispose de logiciels pour effectuer cela dans une étape préliminaire à l'acquisition. Cependant, l'expérience pratique montre qu'un calibrage n'est valide que pour une durée limitée et que les méthodes de calibrage standards, qui nécessitent la mise en place d'objets de référence dans la scène, sont donc peu adaptées pour des recalibrage fréquents. L'idée introduite ici est donc d'explorer la possibilité d'utiliser les silhouettes, qui sont disponibles naturellement dans le processus de modélisation, pour effectuer le calibrage des caméras, et plus particulièrement pour remettre à jour ce calibrage. Les travaux correspondants apparaissent dans [Boy 06] page 65.

Utiliser les silhouettes pour obtenir des informations sur la configuration des caméras n'est pas une idée nouvelle ; Rieger, en 1986, a identifié des points qui appartiennent à plusieurs contours d'occultations sur la surface, *les points fixes*, et propose d'utiliser ces points pour déterminer la rotation de la caméra. Ces points furent par la suite appelés points frontières [Cip 95] et plusieurs méthodes en font usage pour déterminer le calibrage des caméras [Jos 95, Sin 04, Fur 04]. Néanmoins, les points frontières sont difficiles à localiser dans les images et ne fournissent que des contraintes locales sur les cônes de vue. Une autre direction est suivie dans les travaux introduits ici et repose sur l'idée que les cônes de vue d'une même scène s'intersectent intégralement dans l'espace lorsque le calibrage est exact. Une idée similaire apparaît dans [Her 06] où la cohérence entre la projection de l'enveloppe visuelle dans une image et la silhouette dans cette image est considérée pour le calibrage de tables tournantes.

Dans la suite du chapitre, les contraintes fournies par un ensemble de silhouettes sont tout d'abord discutées et des critères pratiques pour le calibrage sont ensuite étudiés.

3.1 La cohérence géométrique des silhouettes

Le calibrage à l'aide de silhouettes repose sur un principe simple : toutes les lignes de vue des silhouettes sont issues d'une même scène et donc tous les cônes de vue doivent s'intersecter de manière exacte dans l'espace. Ce principe de *cohérence géométrique* s'applique lorsque les silhouettes ainsi que le calibrage sont exacts. Dans un processus de calibrage, l'hypothèse faite est que les silhouettes sont exactes ; la cohérence géométrique que les cônes de vue doivent alors vérifier sert à estimer les caractéristiques des caméras. Dans les images, cette cohérence géométrique se caractérise par le fait que la projection de l'enveloppe visuelle, ou l'intersection des cônes de vue, recouvre entièrement les silhouettes. La figure 3.1 illustre ce principe.

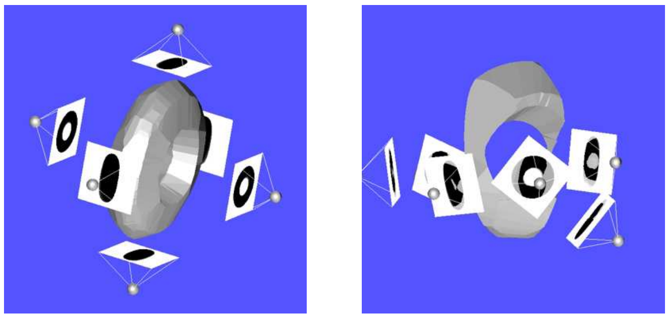


Fig. 3.1 – La cohérence géométrique des cônes de vue : les projections de l'enveloppe visuelle (en noir) recouvrent les silhouettes (en gris) en totalité lorsque le calibrage est exact (à gauche) ; partiellement seulement dans le cas contraire (à droite).

Une première remarque concernant le principe énoncé précédemment est qu'il n'identifie pas nécessairement de façon unique un calibrage. En effet, plusieurs configurations des caméras peuvent éventuellement générer le même ensemble de silhouettes, c'est le cas notamment lorsque l'objet observé présentent des symétries : une sphère par exemple a la même silhouette suivant un nombre infini de points de vue. Néanmoins, les symétries parfaites n'existent pas avec les objets réels et la considération de plusieurs objets simultanément, ou d'un même objet à différentes positions dans une séquence d'images, limite ces situations. Une deuxième remarque est que la cohérence géométrique dont il est question est globale dans le sens où l'ensemble des silhouettes sont impliquées. Elle a, par contre, des conséquences plus locales, entre 2 cônes de vue en particulier ainsi qu'aux points particuliers que constituent les points frontières.

3.1.1 La contrainte de tangence des cônes de vue 2 à 2

Si l'on considère les cônes de vue de la même scène 2 à 2, alors la cohérence géométrique implique que toutes les lignes de vue d'un cône intersectent l'autre et *vice-versa*. Cette contrainte de tangence des cônes 2 à 2 est illustrée dans la figure 3.2.

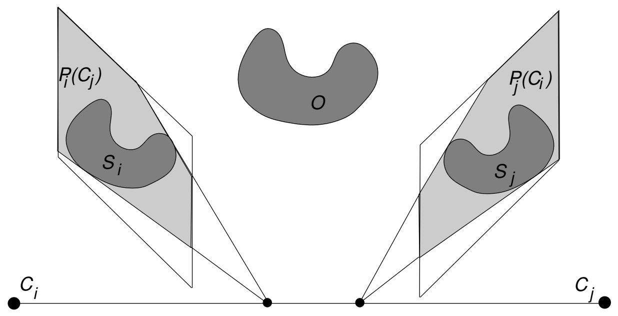


FIG. 3.2 – Dans l'image i, à gauche, la silhouette $\mathcal{S}_\rangle$ est incluse dans la projection $P_i(\mathcal{S}_|)$ du cône associé à la silhouette $\mathcal{S}_|$ et inversement dans l'image j à droite.

Cette contrainte à l'intérêt d'être facile à vérifier dans les images. En revanche, pour un nombre n élevé d'images la complexité en $O(n)$ du nombre de contraintes peut s'avérer rédhibitoire. Un autre aspect important ici est de savoir si le respect de ces contraintes locales par paires de cônes implique en retour le respect de la contrainte globale de cohérence géométrique ? la réponse est que les 2 contraintes ne sont pas exactement équivalentes ; la différence provient du fait qu'avec la cohérence géométrique globale, les cônes doivent non seulement être tangents 2 à 2 mais présenter en plus une intersection commune dans l'espace.

3.1.2 Le lien avec les points frontières

Les points frontières sont des points particuliers où les surfaces de 2 cônes sont tangentes. Ces points appartiennent à la surface de l'objet observé ainsi qu'a l'enveloppe visuelle (cf. 2). Leur intérêt pour le calibrage de caméras provient du fait qu'ils représentent des points sur la surface de l'objet pour lesquels 2, ou plus, projection images sont disponibles. La contrainte qui en découle, la contrainte épipolaire généralisée[Cip 95], est que les droites épipolaires de ces points doivent être tangentes à la silhouette, comme cela est illustré figure 3.3. De nombreuses méthodes de calibrage en font usage, notamment [Jos 95, Sin 04, Fur 04], néanmoins la localisation des points frontières dans les images reste une opération difficile.

Le lien avec les contraintes évoquées précédemment est que la cohérence géométrique implique la présence de points frontières et le respect de la contrainte épipolaire généralisée en ces points. Pour la contrainte de tangence des cônes, cette dernière est équivalente au respect de la contrainte épipolaire généralisée à un sous-ensemble des points frontières : ceux qui appartiennent aux enveloppes convexes des silhouettes. Réciproquement, et de la même manière que pour la contrainte de tangence des cônes, les points frontières lient les images 2 à 2

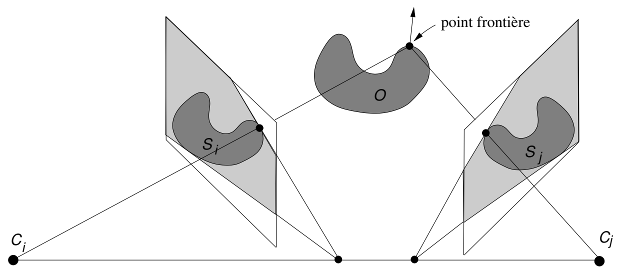


FIG. 3.3 – Les points frontières sont les points qui se projettent sur 2, ou plus, contours de silhouettes. Une fois identifiés dans les images, ils fournissent des contraintes sur la configuration des caméras.

et ne constituent donc pas une équivalence exacte de la cohérence géométrique globale.

3.2 Les critères de calibrage

Les cônes de vue associés à un ensemble de silhouettes doivent vérifier certaines contraintes géométriques comme cela a été vu dans la partie précédente. Ces contraintes géométriques sont satisfaites lorsque le calibrage des caméras est exact, sous les hypothèses que les silhouettes sont exactes et qu'une seule configuration des caméras produit le jeux de silhouettes considéré. Les critères de calibrage qui en découlent évaluent le respect de ces contraintes et peuvent être d'ordres différents : quantitatif ou qualitatif. Les critères quantitatifs sont des fonctions de distances dont l'objectif principal est d'être utilisées dans des processus d'optimisation. Une distance reste par contre relative à une métrique et son interprétation n'est pas toujours aisée. Il existe donc aussi des critères qualitatifs dont la signification est plus évidente qu'une distance mais dont l'usage dans un processus d'optimisation peut s'avérer délicat.

3.2.1 Critères quantitatifs

Un critère quantitatif correspond à une mesure de distance et plusieurs possibilités s'offrent à nous si l'on considère les critères de cohérence évoqués précédemment. Pour la cohérence globale, une possibilité est de mesurer le recouvrement entre la projection de l'enveloppe visuelle et la silhouette : la différence entre les 2 surfaces peut par exemple servir de mesure. Cette idée est utilisée dans [Her 06] ou les pixels des silhouettes n'appartenant pas à la projection de l'enveloppe visuelle sont dénombrés dans les images. Ce critère s'avère en pratique plus qualitatif que quantitatif et nous reviendrons dessus dans la partie suivante. Une autre mesure possible basée sur les points frontières consiste à évaluer la cohérence épipolaire en ces points : la distance entre un point frontière et la droite épipolaire de son correspondant dans une autre image par exemple. Ce critère évalue la tangence des cônes 2 à 2 et non directement la cohérence globale.

Dans [Boy 06] page 65, nous proposons un critère quantitatif qui évalue la distance entre 2 cônes. Cette distance est basée sur celle entre une ligne de vue et un cône. L'intérêt par rapport aux critères mentionnés plus haut est qu'il s'agit bien d'une distance, donc d'une fonction continue et différentiable facilement utilisable dans une optimisation, et que le critère ne dépend pas des points frontières et ne nécessite donc pas leurs localisations. La mise en oeuvre se fait simplement, par une approche des moindres carrés. Ce critère reste associé à la tangence des cônes 2 à 2 et non à la cohérence globale des silhouettes. Néanmoins, mentionnons ici que la cohérence globale se prête mal aux critères quantitatifs car toutes les configurations ou les cônes de vue ne définissent pas d'enveloppe visuelle sont considérées comme équivalentes, or il s'agit là d'une vaste majorité des configurations. En revanche, la cohérence globale débouche sur des critères qualitatifs qui permettent d'évaluer finement un calibrage comme nous le verrons dans la partie suivante. La figure 3.4 illustre le principe de l'optimisation d'un calibrage à l'aide de la distance entre cônes de vue.

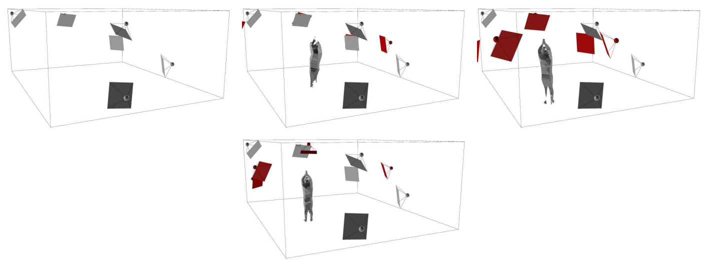


FIG. 3.4 – Une illustration de la minimisation des distances entre cônes de vue. De gauche à droite : la configuration initiale des caméras pour laquelle il n'y a pas d'enveloppe visuelle ; en rouge les configurations intermédiaires des caméras obtenues à des itérations successives de l'algorithme d'optimisation.

Dans l'exemple ci-dessus, le calibrage des caméras s'obtient par optimisation des paramètres des caméras. Une question cruciale est alors comment obtenir des valeurs initiales pour ces paramètres ? comme élément de réponse, notons que dans de fréquentes situations, ces valeurs initiales sont disponibles, c'est le cas notamment de beaucoup de plateformes d'acquisition où les caméras sont fréquemment aux mêmes positions. Par ailleurs, notons aussi que les points frontières permettent d'obtenir de telles valeurs même si leur localisation est imparfaite.

3.2.2 Critères qualitatifs

Les critères quantitatifs ne permettent pas toujours d'interpréter un résultat. Dans le cadre du calibrage à partir de silhouettes, nous avons donc proposé des critères qui permettent d'évaluer la qualité combinée d'un ensemble de silhouettes et d'une configuration de caméras. Ces critères sont tous basés sur la cohérence globale des silhouettes et donc sur le fait que les cônes de vue associés aux silhouettes définissent une enveloppe visuelle et que cette dernière recouvre complètement, après projection dans les images, les silhouettes. Le premier critère est celui mentionné précédemment, et par ailleurs proposé dans [Her 06], qui consiste à compter dans les silhouettes les pixels appartenant à la

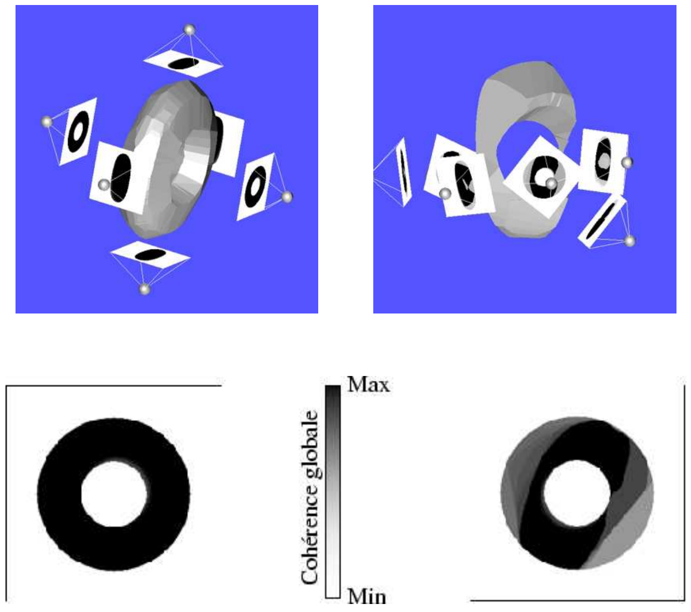


FIG. 3.5 – La mesure de la cohérence globale des silhouettes par pixel dans les images : à gauche les silhouettes et la configuration des caméras sont exactes, à droite la configuration des caméras a été modifiée.

projection de l'enveloppe visuelle. Ce critère évalue bien la cohérence globale des silhouettes mais présente le désavantage d'un niveau de discrétisation peu précis ; pour chaque pixel, la cohérence globale est représentée par 2 niveaux seulement, correspondant au fait que le pixel soit à l'intérieur ou non de la projection de l'enveloppe visuelle. Une amélioration naturelle de ce critère consiste à compter, pour chaque pixel, le nombre de cônes qu'intersecte sa ligne de vue. Pour n caméras, ce nombre doit être $n-1$. D'autres améliorations sont possibles pour affiner encore plus le critère, comme cela est précisé dans [Boy 06] page 65 et illustré dans la figure 3.5

Chapitre 4

Estimation des formes visuelles

Une partie centrale des travaux présentés dans ce mémoire concerne l'estimation des formes observées à partir de plusieurs point de vue, ces formes pouvant évoluer dans le temps. Ce thème constitue un axe de recherche important de la communauté vision depuis plusieurs décennies. Plusieurs raisons expliquent cet engouement dont notamment le fait que le processus de vision humain permet de percevoir les formes tridimensionnelles et que cela enrichit de manière considérable notre perception du monde. Du point de vue de la vision artificielle, l'intérêt de la modélisation tridimensionnelle est double ; il s'agit bien sur de produire de manière automatique des modèles réalistes de notre environnement mais aussi d'obtenir par ce biais une représentation compacte qui englobe tous les points de vue d'un objet ou d'une scène, et permet donc de reproduire ces points de vue à volonté. Les applications qui en découlent couvrent un éventail assez large : la navigation, la modélisation, la reconnaissance et plus particulièrement dans le cadre des travaux présentés ici, les interactions entre mondes virtuels et réels.

L'estimation de formes à partir de plusieurs vues peut se faire à l'aide de différentes primitives dans les images : des points, des contours ou des régions. Nous considérons dans ce document les silhouettes qui sont les régions dans l'image correspondant aux objets d'intérêts d'une scène. Dans le cadre de l'estimation de formes, les silhouettes présentent de sérieux avantages par rapport aux autres primitives. À la différence des points ou contours, les silhouettes ne souffrent pas des occultations et ne sont pas dépendantes de critères, souvent instables, comme la cohérence photo-métrique. Elles peuvent toujours être extraites dans les images, par des techniques standard de soustraction de fond notamment. Par ailleurs, un ensemble de silhouettes définit directement une forme dans l'espace, une surface ou un volume, et non une représentation discrète de cette forme comme cela est le cas avec des primitives de types points ou contours. De plus, de nombreuses méthodes relativement simples à implémenter sont apparues pour estimer cette forme. Pour ces raisons, les silhouettes sont très largement utilisées dans la communauté vision.

Parmi les premiers travaux de modélisation à partir de silhouettes, ceux de Baumgart en 1974 [Bau 74] constituent une contribution majeure où les principaux éléments de ce type de modélisation sont identifiés : les silhouettes dans les images définissent par rétro-projection des régions dans l'espace à l'intérieur desquels se trouve les objets d'intérêts et Baumgart propose de calculer l'intersection de ces régions à l'aide des opérateurs d'Euler. Cette intersection définit un volume dans l'espace qui englobe les objets à modéliser de manière plus ou moins précise en fonction du nombre de point de vue considéré. Ce volume approximant sera par la suite communément appelé *enveloppe visuelle*, à l'initiative de Laurentini[Lau 94]. Depuis Baumgart, les silhouettes ont été très largement utilisées dans les communautés de la vision par ordinateur et du graphisme. Les méthodes de calcul qui en découlent suivent deux directions distinctes : une direction déterministe où les silhouettes et le calibrage des caméras sont supposés exactes ; et une direction probabiliste où des incertitudes sont introduites dans ces données. La structure de ce chapitre reflète cet aspect. Les approches déterministes que nous proposons ont été motivées par le besoin de méthodes de modélisation précises et rapides, notamment dans le cadre d'application interactive et donc temps réel. Elles sont présentées dans la première partie de ce chapitre. Ces méthodes présentent en revanche une forte dépendance à la qualité des silhouettes dans les images et du calibrage des caméras. Pour remédier à cela, nous avons aussi étudié une approche probabiliste dont les principes sont introduits dans la deuxième partie du chapitre. Enfin, la troisième partie du chapitre traite de l'implémentation distribuée des algorithmes déterministes sur une grappe de PC en vue d'applications interactives.

4.1 Approches déterministes

Les approches déterministes reposent sur l'hypothèse que les silhouettes, ainsi que le calibrage des caméras, sont exacts. Cette hypothèse est bien sur très forte, néanmoins elle s'avère réaliste lorsque l'environnement d'acquisition est contrôlé, soit lorsque l'environnement est statique et d'aspect de préférence uniforme, et lorsque les caméras sont fixes. Les approches existantes se classent ensuite en deux catégories principales, les approches volumiques et les approches surfaciques, à laquelle s'ajoute une approche purement image proposée par Matusik *et al.* [Mat 00] qui n'estime pas l'enveloppe visuelle mais ces projections images. Les approches volumiques considèrent une discrétisation de l'espace en cellules élémentaires appelés *voxels* (*volume élément* par analogie avec les *pixels*). L'espace discret ainsi obtenu est ensuite sculpté de façon à ne conserver que les voxels appartenant à l'enveloppe visuelle, en d'autres termes, les voxels se projetant à l'intérieur de toutes les silhouettes considérées. Dans cette catégorie, une première approche de Martin et Aggarwal [Mar 83] utilisait des cellules parallélépipédiques alignées avec les axes de coordonnées. Par la suite, des améliorations furent proposés, notamment les *octrees* [Chi 86] et des méthodes de calcul efficaces [Sze 93, Nie 94, Che 00]. Les approches reposant sur une discrétisation de l'espace présentent un avantage certain qui est la robustesse. En revanche, elles sont clairement limitées par le compromis entre précision et complexité qui découle de la discrétisation.

Les approches surfaciques suivent une stratégie différente, les éléments de la surface de l'enveloppe visuelle sont estimées en intersectant les cônes de vue associés aux silhouettes. C'est la direction que prend initialement Baumgart [Bau 74], bien avant les discrétisations voxéliques rendues possibles par l'extension des capacités de mémoire des ordinateurs. Cette direction sera ensuite reprise à la fin des années 90 en raison de la précision du résultat qui peut être obtenue[Sul 98]. Baumgart utilisait à l'origine les opérateurs d'Euler pour effectuer l'intersection deux à deux des cônes de vue polyédriques associés à des silhouettes polygonales dans les plans images. Dans la même veine, les représentations *Brep (Boundary representation)* peuvent être utilisées pour effectuer ce calcul [Li 02], ainsi que les représentations *CSG (Constructive Solid Geometry)* [Isi 02]. Ces dernières approches ont le mérite de la simplicité : des librairies standards du domaine public, CGAL [CGA] par exemple, peuvent réaliser cette opération. En revanche, elles n'utilisent pas la spécificité du problème, à savoir que les cônes de vue ne sont pas des polyèdres généraux mais des polyèdres infinis issus de la rétro-projection de silhouettes planes, avec pour conséquence pour ces approches peu de robustesse et donc de fréquentes situations sans résultat, ainsi que des temps de calcul dissuasifs pour de nombreuses applications.

L'enveloppe visuelle constitue une boite englobante de l'objet, ou de la scène, observé. Partant de cette constatation, plusieurs travaux proposent d'estimer une surface à laquelle l'enveloppe visuelle est tangente. Dans [Cip 92, Vai 92, Boy 95], des premières solutions locales sont proposées ; elles permettent de déterminer des points de contact entre surface et enveloppe visuelle le long des lignes de vue des sommets des silhouettes. D'autres approches [Cro 98, Kan 01, Bra 04, Lia 05] exploitent le principe de dualité entre points et plans dans l'espace 3D et estiment la surface localement duale de l'enveloppe visuelle, c'est à dire la surface ayant un point de contact dans chaque face de l'enveloppe visuelle. Les approches qui estiment une surface tangente à l'enveloppe produisent de meilleures approximations que celle que constitue l'enveloppe visuelle. Néanmoins, les travaux mentionnés n'exploitent pas l'ensemble des informations fournies par les silhouettes et souffrent de singularités fréquentes qui rendent leur exploitation difficile.

Comme cela a été vu dans le chapitre des définitions, nous regroupons les notions d'enveloppe visuelle et de surface duale de l'enveloppe visuelle sous une seule et même définition, celle des formes visuelles, dont l'intérêt est de fournir un cadre unifié pour la description de formes 3D dont les projections images correspondent à un ensemble donné de silhouettes. Pour estimer ces formes, nous avons développé des méthodes appartenant à la catégorie des approches surfaciques en raison de leur précision et de leur rapidité. Ces méthodes calculent les intersections des lignes de vue des sommets des silhouettes avec les formes visuelles. Ces intersections sont ensuite maillées à l'aide de la triangulation de Delaunay pour produire une surface constituée de facettes triangulaires. Dans le cas de l'enveloppe visuelle, la surface ainsi obtenue est proche de l'intersection des cônes de vues sans être exactement cette dernière du fait que les intersections ne sont pas calculées sur toute la surface des cônes de vue mais uniquement pour les lignes de vue des sommets des silhouettes. Dans le cas de l'enveloppe visuelle, il est possible de déterminer la surface de manière exacte sans prendre en compte l'ensemble des lignes de vue des cônes et nous avons proposé une méthode dans ce sens.

4.1.1 Les formes visuelles par triangulation de Delaunay

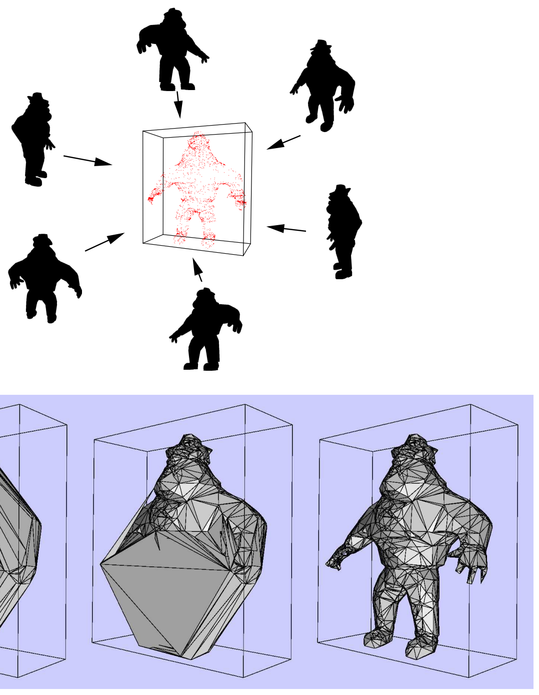


Fig. 4.1 – Les différentes étapes de l'estimation des formes visuelles à l'aide de la triangulation de Delaunay : en haut le calcul de points sur la surface à partir des silhouettes ; en bas la triangulation de Delaunay de ces points et la sculpture de cette triangulation.

Dans cette partie, nous introduisons la méthode d'estimation des formes visuelles dont l'origine se trouve dans l'article [Boy 03], page 48, dans le cas de l'enveloppe visuelle et dont la généralisation aux formes visuelles se trouve dans l'article [Lap 06], page 56. Les principales étapes de cette méthode sont illustrées dans la figure 4.1.

Les formes visuelles associées à un ensemble de silhouettes sont, par définition, incluses dans les cônes de vue des silhouettes et tangentes à ces cônes. Une première étape commune à l'ensemble des approches déterministes que nous proposons va donc consister à calculer les intersections des surfaces des cônes de vue avec les formes visuelles. Comme cela a été mentionné précédemment, ces intersections ne sont pas calculées sur l'ensemble de la surface d'un cône mais uniquement le long des lignes de vue des sommets des silhouettes. Ces intersections, ou contribution des lignes de vue, sont appelés *segments de vue* dans le cas de l'enveloppe visuelle (figure 4.2). Pour les autres formes visuelles, et comme l'enveloppe visuelle contient l'ensemble des ces formes, il est facile de déduire que les contributions des lignes de vue sont alors incluses dans les segments de vue.

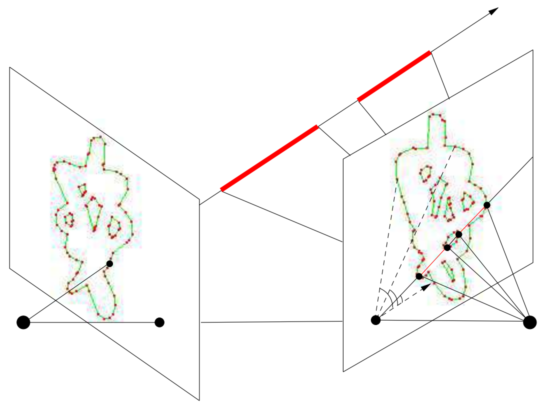


Fig. 4.2 – Les segments de vue le long de la ligne de vue pour 2 silhouettes.

La détermination des ces segments de vue est une opération simple qui consiste à intersecter, dans les images, les droites épipolaires avec les silhouettes, puis à conserver la partie commune des intervalles ainsi obtenus (voir [Boy 03] page 48 pour plus détail). Cette opération peut être effectuée rapidement comme cela sera discuté dans la partie implémentation temps réel de ce document.

Contributions des lignes de vue aux formes visuelles

Dans le cas général, l'intersection d'une ligne de vue avec la forme visuelle correspond à un sous-ensemble des segments de vue le long de cette ligne. Ce sous-ensemble peut aller du segment de vue dans son intégralité pour l'enveloppe visuelle, jusqu'au point de contact unique pour une surface à laquelle l'enveloppe visuelle est tangente. Dans ce dernier cas, la position du point de contact à l'intérieur du segment visuel ne peut être déterminée sans hypothèse supplémentaire. Nous supposons que le la surface recherchée est localement d'ordre 2, dans un voisinage délimité par les extrémités du segment de vue. La position du point de contact peut alors être déterminée de manière linéaire, comme nous l'avons montré dans des travaux antérieurs [Boy 97].

Les approches duales mentionnées précédemment [Cro 98, Kan 01, Bra 04, Lia 05] calculent aussi localement un point sous l'hypothèse d'une surface localement lisse. L'idée est, pour un sommet d'une silhouette, d'estimer localement la surface à l'aide des plans tangents fournis par : les voisins du sommet sur la silhouette où il se trouve ; les voisins sur 2 silhouettes proches. Le voisinage ainsi constitué ne permet pas d'assurer que le point de contact sera sur l'enveloppe visuelle, dans la pratique il en sera même fréquemment très éloigné. C'est la que réside une importante différence avec notre approche qui assure que le point de contact sera bien à l'intérieur du segment de vue et qui considère le meilleur voisinage possible pour l'estimation locale de la surface et de son point de contact, celui défini par les points extrémités du segment de vue. Une autre différence est que notre approche traite indifféremment les scènes quelle que soit leur topologie. Par ailleurs toutes les formes visuelles associées à un ensemble donné de silhouettes ont, par construction, la même topologie et donc celle de l'enveloppe visuelle.

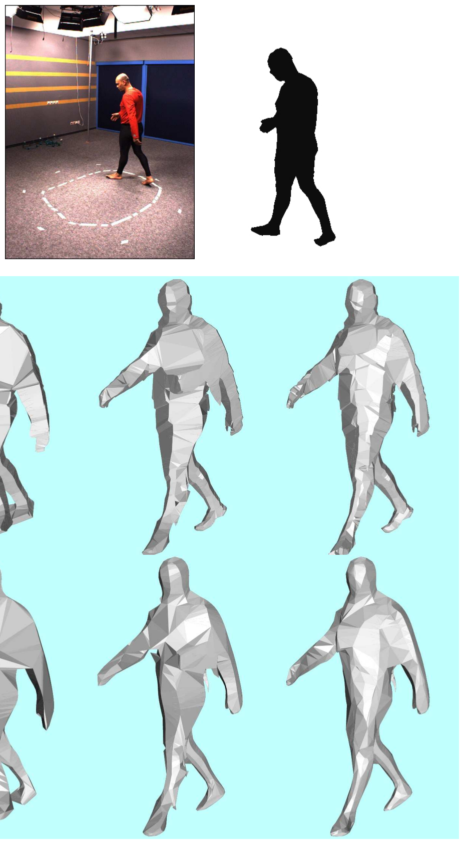


Fig. 4.3 – les formes visuelles d'un corps humain pour, de gauche à droite, 2, 4 et 6 silhouettes : au milieu l'enveloppe visuelle ; en bas la forme visuelle ne possédant qu'un point de contact avec les segments de vue.

Triangulation des contributions

Une fois les contributions le long des lignes de vue des sommets des silhouettes déterminées, il reste à construire un maillage ou une surface qui lie ces contributions. Nous utilisons la triangulation de Delaunay pour cela. Les données d'entrée sont les points extrémités des segments de contributions le long des lignes de vue : les extrémités des segments de vue dans le cas de l'enveloppe visuelle et le point de contact unique dans le cas de la surface duale de l'enveloppe visuelle. La triangulation de Delaunay dans $\mathbb{R}^3$ de ces points fournis alors un ensemble de tétraèdres (figure 4.1) qui vont ensuite être filtrés pour éliminer ceux qui se projettent à l'extérieur d'une, ou plus, silhouette. Les tétraèdres restant constituent la forme visuelle résultat et leur frontière extérieure sa surface. La figure 4.3 montre des exemples de formes visuelles pour différentes contributions le long des lignes de vue.

4.1.2 L'enveloppe visuelle par une approche exacte

Les méthodes présentées dans la partie précédente calculent une approximation de la surface de la forme visuelle. Cette approximation résulte, en premier lieu, du fait que l'ensemble des points constituant le contour des silhouettes ne sont pas pris en considération mais uniquement les sommets des silhouettes polygonales ; en deuxième lieu, les jonctions entre ces contributions, ou leur maillage, ne sont pas définies de façon exacte et nous supposons qu'elles appartiennent à la triangulation de Delaunay. Le cas de l'enveloppe visuelle représente alors une exception, sa surface est entièrement constituée de contributions le long des lignes de vue issues des contours des silhouettes et est donc parfaitement définie. Dans le cas de silhouettes polygonales, cette surface est celle du polyèdre intersection de l'ensemble des cônes de vue associés aux silhouettes considérées et peut être déterminée de manière exacte, comme nous l'avons expliqué dans [Fra 03].

L'approche que nous introduisons ici, [Fra 03], page 76, s'appuie sur la structure de l'enveloppe visuelle. Celle-ci est composée uniquement de morceaux de cônes de vue, les bandes, ou *strips*. Ces bandes contiennent les segments de vue des sommets des silhouettes (figure 4.2) qui en constituent une description partielle. L'idée est alors de compléter les segments de vue pour reconstituer les bandes dans leur intégralité. L'algorithme de calcul de l'enveloppe visuelle se décompose donc en trois étapes principales, illustrée figure 4.4 :

1. Le calcul des segments de vue (décrit précédemment).
2. Les segments de vue sont complétés en parcourant, à partir des extrémités des segments, les courbes d'intersection des cônes de vue. On ajoute de cette manière des segments de courbes d'intersection de cônes ainsi que des points triples lorsque 2 courbes, en d'autre terme 3 cônes, s'intersectent
3. Le maillage déterminé est parcouru pour déterminer les facettes qui composent l'enveloppe visuelle.

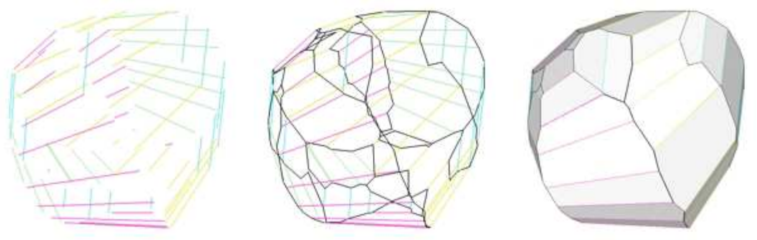


Fig. 4.4 – Les 3 étapes du calcul de la surface de l'enveloppe visuelle d'une sphère (et pour 3 cônes de vue) : (a) les segments de vue ; (b) le maillage ; (c) les facettes polygonales.

L'étape 2 de l'algorithme d'identification du maillage se base sur la propriété remarquable que l'enveloppe visuelle polyédrique ne contient, dans les situations génériques, que des sommets de valence 3. Cela signifie que localement le voisinage d'un sommet est entièrement déterminé lorsque ces 3 sommets voisins sont identifiés. L'intérêt est de rendre l'algorithme efficace au sens où les calculs sont minimisés et consistent à intersecter des segments en 2D. Néanmoins, les algorithmes de type CSG évoqués précédemment peuvent présenter des complexités théoriques équivalentes à celui introduit ici. L'expérience prouve en revanche que le calcul d'intersection de polyèdres, sur lequel débouchent les approches CSG, est difficile à mettre en oeuvre de manière robuste et rapide.

4.2 Une approche probabiliste pour l'enveloppe visuelle

Les données nécessaires au calcul des formes visuelles, et de l'enveloppe visuelle en particulier, sont les silhouettes et le calibrage des caméras d'acquisition des silhouettes. Ces données sont en pratique entachées d'erreurs, en raison notamment du bruit présent dans toute la chaîne d'acquisition et de la difficulté d'extraire des silhouettes précises. C'est pourquoi des approches probabilistes prenant en compte les incertitudes sur ces données ont vu le jour. Snow *et al.* [Sno 00] ont notamment proposé une approche dans laquelle la décision de présence d'un objet d'intérêt ne se fait pas au niveau du pixel dans les images, comme cela est le cas avec les approches utilisant les silhouettes, mais de manière globale pour des voxels dans l'espace d'acquisition. L'idée est de minimiser une énergie prenant en compte, pour chaque voxel, les informations provenant de l'ensemble des pixels sur lesquels le voxel se projette ainsi que les décisions d'occupation des voxels voisins. Une approche similaire dans le principe, mais non voxélique [Zen 04], propose de déterminer les silhouettes dans les images, non pas de manière individuelle, mais en tenant compte de la cohérence géométrique qui doit exister entre toutes les silhouettes d'une même scène. Ces approches ont le mérite d'intégrer l'information issue de plusieurs images et de minimiser ainsi les erreurs d'imprécisions. Elles ne proposent pas toutefois de modèles d'incertitudes explicites. Une autre direction intéressante a été suivie par Grauman *et al.* [Gra 03a] et consiste à choisir les silhouettes qui correspondent le mieux aux données images dans une base apprise *a priori*. L'avantage est que les silhouettes correspondent alors de manière exacte à un modèle 3D, même avec une calibrage et des données images imprécises, l'inconvénient majeur étant que les modélisations possibles restent limitées par la base d'apprentissage.

Les approches mentionnées estiment des valeurs d'occupation binaires, pour des pixels ou des voxels. Les grilles stochastiques d'occupation améliorent cela dans le sens où une probabilité, au lieu d'une décision, est estimée pour l'occupation. Elles ont été à l'origine proposées dans la communauté robotique pour modéliser l'environnement d'un robot à partir de données issues de capteurs [Mor 85]. Elles se sont avérées particulièrement efficaces pour fusionner des données imprécises provenant de multiples capteurs. Le cas des capteurs images a par ailleurs été traité dans cette communauté, dans un contexte par contre restreint de localisation d'objets [Mar 98]. Des grilles de probabilité ont aussi été utilisées en vision par ordinateur, [Bro 01] par exemple, mais principalement pour estimer des cohérences photo-métriques, ce qui s'avère être un problème plus complexe que celui que nous cherchons à résoudre ici.

L'approche que nous décrivons succinctement ici, [Fra 05] page 87, utilise les grilles d'occupation pour modéliser la scène. Dans cette approche, les images sont considérées comme des matrices de capteurs où chaque capteur-pixel fournit une information sur la présence d'un objet le long de sa ligne de vue. L'ensemble de ces informations est alors fusionné dans la grille d'occupation pour fournir une probabilité en chaque voxel.

4.2.1 La formulation Bayésienne du problème

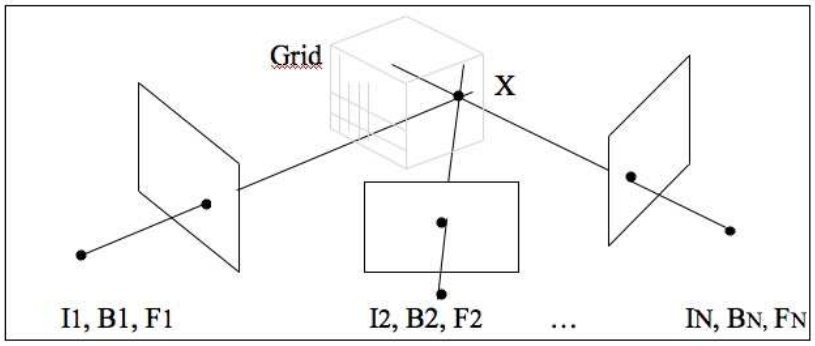


Fig. 4.5 – La probabilité X d'un voxel de la grille est déterminée à partir des intensités I des pixels à un instant donné, du modèle B du fond pour ces pixels et des variables binaires cachées F d'appartenance des pixels à la silhouette.

La scène observée est supposée être composée d'objet d'intérêts situés devant un fond statique. Les données du problème sont, en dehors du calibrage des caméras, les images à un instant donné et les données apprises du fond, sous la forme d'un modèle statistique pour les intensités d'un pixel des images du fond, une distribution Gausienne par exemple. Chaque pixel représente la réponse bruitée d'un capteur à l'ensemble des voxels de la grille d'occupation recherchée et la résolution du problème nécessite la modélisation des relations entre ces variables. Nous avons choisi le formalisme de Bayes dans lequel ces relations sont régies par une loi de probabilité conjointe à partir de laquelle nous allons, par inférence, déterminer la vraisemblance d'occupation de chaque voxel. Les aspects importants de cette modélisation sont que pour rendre le problème soluble certaines interdépendances entre variables ne sont pas considérées. Nous faisons notamment l'hypothèse que les voxels sont statistiquement indépendant, comme cela est classique avec les grilles d'occupation. Un deuxième aspect important est que pour permettre la prise en compte d'incertitudes dans le processus de formation des images, nous introduisons un jeu de variables cachées qui caractérisent la présence ou non d'un objet d'intérêt devant chaque pixel. Ces variables d'état binaires, une pour chaque pixel, caractérise l'appartenance d'un pixel à une silhouette et permettent de modéliser les incertitudes sur le calibrage : plusieurs voxels peuvent expliquer l'état d'un pixel, et sur les intensités dans l'image : par exemple les intensités du fond et du pixel courant peuvent être similaire alors que le pixel appartient à la silhouette. La résolution du problème se fait ensuite en marginalisant la probabilité conjointe. Les détails de la modélisation sont données dans [Fra 05] page 87. Les figures 4.6 et 4.7 montrent quelques résultats. La figure 4.7 illustre notamment l'intérêt d'une approche multi-images pour améliorer la soustraction de fond.

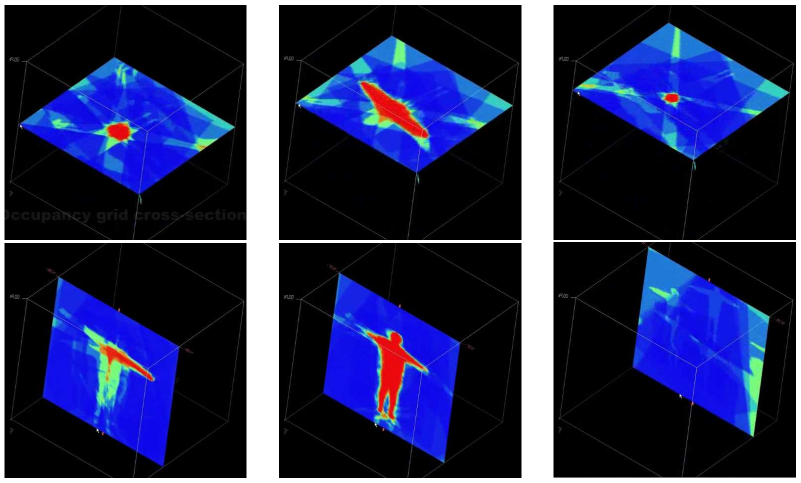


Fig. 4.6 – Un exemple de grille d'occupation vue en coupes horizontales (haut), et verticales (bas). En rouge les zones à fortes probabilités, en vert les zones équiprobables, en bleu les zones à faibles probabilités.

4.3 Implémentations temps réel

Une partie importante des travaux présentés dans ce document a été développée dans un contexte pratique, celui de la plateforme Grimage, avec pour objectif de permettre les interactions entre mondes réels : un acteur par exemple, et mondes virtuels : un environnement constitué d'objets virtuels. Cet objectif nécessite la mise en oeuvre de méthodes temps-réel, en particulier pour la modélisation 3D présentée dans ce chapitre. Le développement de ces méthodes temps-réel requiert un travail spécifique pour que les algorithmes de modélisation, même s'ils sont à l'origine rapides, puissent fonctionner avec des contraintes fortes, un nombre important de caméras notamment. Dans cette partie, nous introduisons les travaux que nous avons réalisés sur ce thème et qui sont par ailleurs détaillés dans les articles [Fra 04, All 06] pages [95,104]. Ces travaux s'inscrivent dans la lignée de ceux, précurseurs, de l'université de Carnegie Mellon sur la *virtualisation*[Nar 98, Che 00], et de ceux de l'ETH de Zurich autour de l'impressionant projet *Blue C* [Gro 03, Wö4] ainsi que plusieurs approches voxéliques dont [Bor 00, Kam 00, Ari 01, Wu 03]. L'originalité de notre approche par rapport à celles mentionnées réside tout d'abord dans la partie matérielle : nous utilisons un nombre flexible de composants standards interchangeables (caméras, PCs, projecteurs, etc.), puis dans les modèles 3D produits : leur qualité ne dépend pas d'un niveau de discrétisation comme cela est le cas des modèles voxéliques.

2 critères importants interviennent dans la mise en oeuvre de la modélisation en temps réel : le débit et la latence. Le premier caractérise le nombre de modèles qui peuvent être produits en un temps fixe et le deuxième le temps qu'il faut pour produire un modèle. Pour traiter ces 2 aspects, nous avons travaillé à différents niveaux de profondeur dans les méthodes : tout d'abord la distribution des algorithmes sur l'ensemble des ressources de calcul pour contrôler le débit, puis

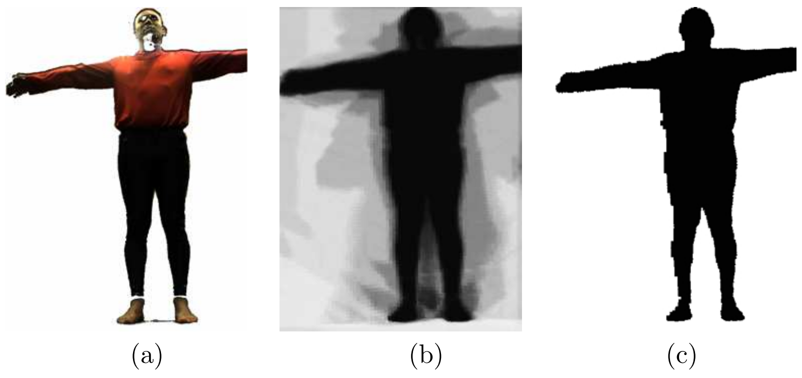


FIG. 4.7 – Une illustration d'application de la grille d'occupation, la soustraction de fond multi-images : (a) une soustraction monoculaire classique ; (b) la projection de la grille dans l'image, l'intensité représente la probabilité d'occupation (noir = 1, blanc = 0) ; (c) la silhouette obtenue par seuillage de l'image en (b)).

la parallélisation au sein des algorithmes pour améliorer la latence.

4.3.1 Le schéma de distribution

La distribution des taches sur l'ensemble des ressources est faite selon le schéma de la figure 4.8. Le principe est que chaque caméra se voit attribuée un PC pour effectuer les pré-traitements des images : l'acquisition et la soustraction de fond. Les PCs restant sont alors répartis entre la modélisation, la gestion de l'affichage ou d'autres taches éventuelles.

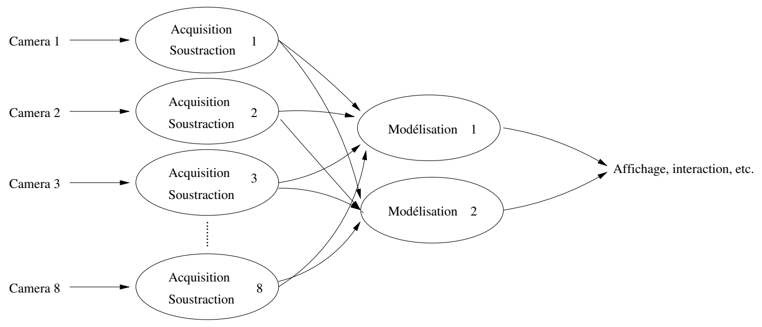

Camera 1
Acquisition Soustraction 1
Camera 2
Acquisition Soustraction 2
Camera 3
Acquisition Soustraction 3
Camera 8
Acquisition Soustraction 8
Modélisation 1
Modélisation 2
Affichage, interaction, etc.

FIG. 4.8 – La configuration à 8 caméras de la plateforme Grimage : chaque ellipse représente un PC.

Dans le schéma de distribution présenté, plusieurs ressources de calcul sont attribuées à l'étape de modélisation, qui est l'étape la plus coûteuse du processus. Cette étape est donc elle-même décomposée et parallélisée de manière à réduire son coût.

4.3.2 La parallélisation des méthodes de modélisation

Les approches considérées ici sont les approches déterministes présentées au paragraphe 4.1. Elles se décomposent en 3 étapes principales pour chaque trame : la détermination des segments de vue ; la détermination du maillage ; l'extraction de la surface. Ces 3 étapes sont consécutives, une étape ne peut en effet démarrer sans les résultats de la précédente. La parallélisation tient compte de cela. Dans un premier temps, et pour contrôler le débit, le traitement des trames est distribué sur les unités de calcul disponibles selon le principe du *pipeline*. Cela permet de démarrer le traitement d'une trame -*pipeline stage*- avant d'avoir terminé celui de la précédente, et donc de contrôler le débit lorsque suffisamment de ressources sont disponibles. La figure 4.9 illustre ce principe pour un algorithme à 2 étapes consécutives A et B.

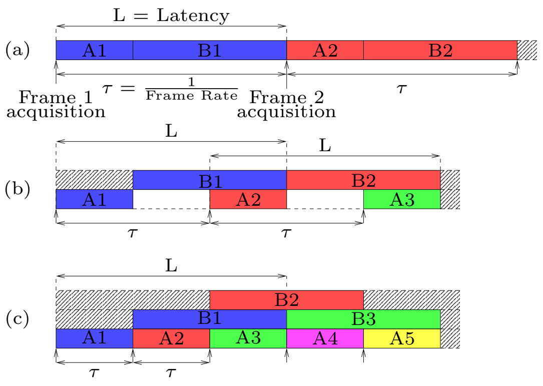


FIG. 4.9 – At et Bt sont 2 étapes du traitement de la trame t. Chaque ligne représente un processeur : (a) une exécution séquentielle, (b)-(c) une exécution à 2-3 processeurs.

Le principe décrit précédemment permet d'augmenter le débit en ajoutant des ressources de calcul. En revanche, il ne modifie pas la latence qui reste égale à la somme des temps d'exécution des différentes étapes. Dans un deuxième temps, chaque étape de l'algorithme a donc elle-même été divisée en sous-étapes non consécutives qui sont alors exécutées en parallèle afin de diminuer le temps d'exécution de l'ensemble de l'étape. Cela a nécessité un travail plus fin de découpage des algorithmes. Par exemple, la détermination des segments de vue se fait de manière indépendante par segment de vue ou groupe de segments de vue. Il est à noter ici que si les algorithmes peuvent être découpés en sous-étapes, l'ensemble des données doit en général rester accessible à toutes les sous-étapes. Une autre remarque concerne la triangulation de Delaunay dont il est fait usage dans certaines approches de modélisation. La triangulation de Delaunay semble en effet réfractaire à la parallélisation et nous n'avons pas trouvé de méthodes ou d'algorithmes fiables pour cela. Cette étape reste donc exécutée de manière séquentielle dans nos implémentations temps réel.

Pour gérer le découpage en étapes, et sous-étapes, de façon modulaires, ainsi que les communications nécessaires entre les différents modules, nous avons utilisé l'intergiciel *flowVR* développé à l'INRIA Rhône-Alpes par l'équipe MOAIS. En termes de résultats, les algorithmes de calcul de l'enveloppe visuelle ont été portés avec succès sur la plateforme Grimage ou ils s'exécutent à 30 trames par seconde avec des latences variables pour les différents algorithmes mais de l'ordre de 50ms pour la méthode exacte, ce qui est raisonnable pour les interactions car pratiquement imperceptible. Concernant la flexibilité, le système temps-réel fonctionne sur la plateforme Grimage avec 8 caméras et 11 PCs ainsi que sur une mini-plateforme comprenant 5 caméras et 6 mini-PC.

Chapitre 5

Estimation du mouvement

Les chapitres précédents traitent de l'acquisition de données géométriques tridimensionnelles à l'aide de systèmes multi-caméras. Ces données donnent accès à de nombreuses applications, dont la génération du nouveau point de vue, mais elles ne permettent pas, dans leur forme brute, la réalisation de tâches plus complexes telles que la manipulation d'objets virtuels par exemple. Ces tâches nécessitent une information complémentaire qui permette l'association entre un objet virtuel et les données acquises. C'est l'objet de la capture de mouvement qui estime le mouvement d'un objet dans le temps, typiquement le mouvement articulaire d'un être humain. Le mouvement estimé peut ensuite être transféré sur un objet virtuel, ou encore interprété pour de la reconnaissance par exemple.

L'estimation du mouvement à l'aide de caméras est un sujet abondamment traité dans les communautés vision et graphique depuis de nombreuses années. Lorsque l'environnement est contrôlable, des solutions robustes existent. En particulier, des systèmes commerciaux fonctionnent avec succès, le système Vicon[1] par exemple. Ces systèmes utilisent un modèle de l'objet en mouvement, typiquement un modèle biomécanique du mouvement humain, et des marqueurs positionnés sur l'objet pour associer le modèle avec les données. En revanche, pour les environnements moins facilement contrôlables, ceux sans marqueurs en particulier, la capture du mouvement reste un problème difficile et un thème de recherche actuel. Les travaux de recherche correspondants suivent principalement 3 directions différentes. Un premier groupe se base sur l'apprentissage et considère l'estimation du mouvement comme un problème de reconnaissance, [Aga 04, Gra 03b] par exemple, avec succès lorsque les mouvements sont assez proches de la ceux de la base d'apprentissage. Un deuxième groupe suppose connu un modèle de l'objet en mouvement et transforme l'estimation du mouvement en celui de mise en correspondance entre modèle et donné. Enfin un troisième groupe ne suppose aucune connaissance *a priori* et estime directement un modèle articulé et son mouvement, [Chu 03] par exemple, avec les limitations que cela implique en pratique puisque la structure du modèle estimé peut varier de façon significative d'un instant à un autre.

Dans les travaux menés autour de la plateforme Grimage sur le mouvement du corps humain, nous avons considéré les approches de la deuxième catégorie où un modèle articulé de l'objet est connu, cela pour 2 raisons principales : nous

[1] `www.vicon.com/`

souhaitons suivre tous les types de mouvements sans restriction, ce qui désavantage les bases d'apprentissage ; nous souhaitons par ailleurs suivre de manière cohérente les mouvements dans le temps, ce qui est le cas lorsqu'un modèle *a priori* est utilisé. Dans ce cadre nous avons exploré 2 directions différentes : une première approche considère un modèle *a priori* qui inclut la forme et le mouvement. Il s'agit d'un modèle articulé constitué d'ellipsoïdes de dimensions supposées connues. Un deuxième approche se focalise sur le mouvement uniquement et considère un modèle articulé sous la forme d'un squelette dont les dimensions sont ici aussi supposées connues. Ces 2 approches ont été publiées respectivement dans [Nis 05] et [M0́6] , pages [112,123], et sont introduites dans les parties suivantes.

5.1 Un modèle épais

La majorité des approches qui supposent connu un modèle pour estimer le mouvement utilisent pour cela des modèles surfaciques ou volumiques de l'objet en mouvement. Ces modèles combinent l'information sur la forme avec celle sur le mouvement. Le modèle peut être constitué de primitives rigides ou déformables : maillages [Car 03a], cylindres généralisés [Sen 03] ou ellipsoïdes [Che 00, Pla 03] par exemple. La mise en correspondance peut se faire avec des données images monoculaires[Smi 01], images multi-vues [Gav 96] ou tridimensionnelles[Che 00].

Dans notre contexte, nous disposons de données tridimensionnelles fournies par le système d'acquisition. Ces données sont les formes visuelles (cf. 2.3.2 et 4.1.1) pour lesquelles les positions des sommets et les normales à la surface en ces sommets sont connues. L'idée développée dans [Nis 05], page 112, est donc d'utiliser ces données surfaciques pour estimer la posture d'un modèle articulé constitué d'ellipsoïdes. L'originalité de l'approche réside dans les données utilisées qui sont tridimensionnelles, sans être issues d'un processus stéréo. L'intérêt d'utiliser des données tridimensionnelles, plutôt que directement des données images, est que la mise en correspondance entre modèle et données s'effectue dans un seul espace avec une seule métrique au lieu de plusieurs espaces images avec des métriques potentiellement incohérentes. Par ailleurs, l'intérêt des formes visuelles par rapport à d'autres données tridimensionnelles telles que les données stéréo est que les formes visuelles ne nécessitent pas de configuration particulière des caméras et peuvent être estimées dès lors qu'au moins 2 caméras sont disponibles.

5.1.1 Le modèle

Le modèle biomécanique que nous utilisons pour décrire le corps humain est constitué de 21 ellipsoïdes (figure 5.1-gauche), dont les dimensions sont supposées connues. Ces ellipsoïdes sont reliés par des joints en rotation qui modélisent les articulations ; le modèle possédant au total 22 degrés de liberté en rotation et 3 en translation/position. Les ellipsoïdes définissent, par *mélange* selon la technique des *metaballs* ou des *objets mous* [Bli 82, Pla 03], une surface implicite qui modélise la forme du corps humain (figure 5.1-droite). C'est cette surface qui sera mise en correspondance avec les données mesurées.

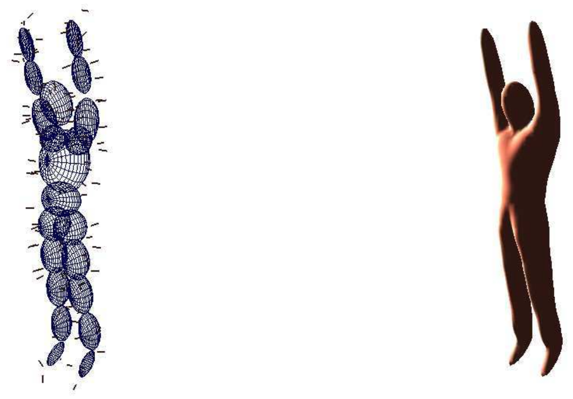


FIG. 5.1 – Le modèle utilisé : à gauche les ellipsoïdes du modèle, à droite la surface implicite associée.

5.1.2 Les données

Les observations auxquelles le modèle ci-dessus est mis en correspondance sont celles issues du processus d'estimation des formes visuelles à partir des silhouettes. Elles se composent des positions dans l'espace des sommets constituant les formes visuelles, ainsi que des normales à la surface en ces sommets. Ces informations sont déterminées pour chaque sommet de chaque silhouette considérée. Les positions dans l'espace sont calculées le long des lignes de vue en supposant la surface observée localement d'ordre 2 (cf. 4.1.1), les normales à la surface sont données par le produit vectoriel de la direction de la ligne de vue et de la tangente à la silhouette. La figure 5.2 montre plusieurs exemples lorsque de 2 à 6 caméras sont utilisées. La figure 5.3 montre 7 trames suivant les 6 points de vue des caméras utilisées.

5.1.3 La mise en correspondance

La mise en correspondance entre les observations et le modèle s'effectue à l'aide d'un estimateur du maximum a posteriori. Cet estimateur maximise la probabilité du modèle sachant les observations en considérant pour cela les distances entres les mesures et le modèle. Un point crucial ici est comment déterminer la distance entre une mesure et le modèle. Cette distance peut être une distance algébrique basée sur la position comme dans [Pla 03]. La fonction que nous utilisons tient compte de la distance à la surface ainsi que de la différence d'orientation entre la normale observée et celle du modèle. La figure 5.4 illustre les résultats obtenus avec cette méthode sur 7 trames et suivant les 6 points de vue des caméras utilisées.

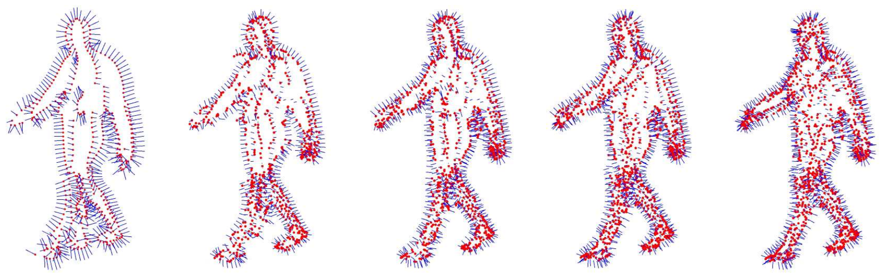


Fig. 5.2 – Les données auxquelles le modèle à base d'ellipsoïdes est mis en correspondance. De gauche à droite, respectivement de 2 à 6 caméras sont utilisées pour l'acquisition de ces données.

5.2 Un modèle mince

En parallèle de l'approche présentée dans la partie précédente, nous avons aussi développé une méthode qui estime le mouvement sous la forme de la pose d'un squelette [Ḿ06], page 123. L'idée principale dans ce travail est de se focaliser sur le mouvement lors de l'estimation sachant que la forme - visuelle - de l'objet observé est disponible (cf. 4.1.1). L'intérêt de découpler le mouvement de la forme est de réduire les hypothèses faites sur le modèle et donc de limiter les erreurs qui en découlent, ce qui a pour conséquences de simplifier la mise en correspondance entre modèle et données et de la rendre plus robuste aux erreurs dans le modèle. Dans le cas du squelette, seules les dimensions des segments ont besoin d'être données. Pour les humains ces dimensions ne sont pas indépendantes mais respectent, en moyenne, des proportions connues, à l'instar du célèbre *homme de Vitruve* de Leonard de Vinci. Peu d'informations *a priori* sont donc ici nécessaires.

Comme cela a été mentionné précédemment, la majorité des approches existantes pour l'estimation du mouvement utilisent des modèles qui incluent la forme et assez peu de travaux en vision par ordinateur se focalisent sur le mouvement. Proche de la méthode présentée ici, [Luc 01, The 02] proposent d'estimer un squelette, mais les modèles considérés possèdent des caractéristiques volumétriques ; Brostow *et al.* [Bro 04] proposent une approche basée sur les squelettes mais dont l'objectif est d'estimer le modèle lui-même et non le mouvement d'un modèle *a priori*. La contribution du travail présenté dans [Ḿ06] est donc une approche qui s'affranchit en grande partie des paramètres de formes, en combinant un modèle du squelette humain avec l'axe médian de la forme visuelle obtenue à partir des silhouettes. Les paragraphes suivant introduisent cette approche, des détails se trouvent dans [Ḿ06], page 123.

5.2.1 Le modèle

Le modèle biomécanique qui est utilisé ici est constitué de 12 segments (figure 5.5), dont les longueurs sont supposées connues. En pratique, la mise en

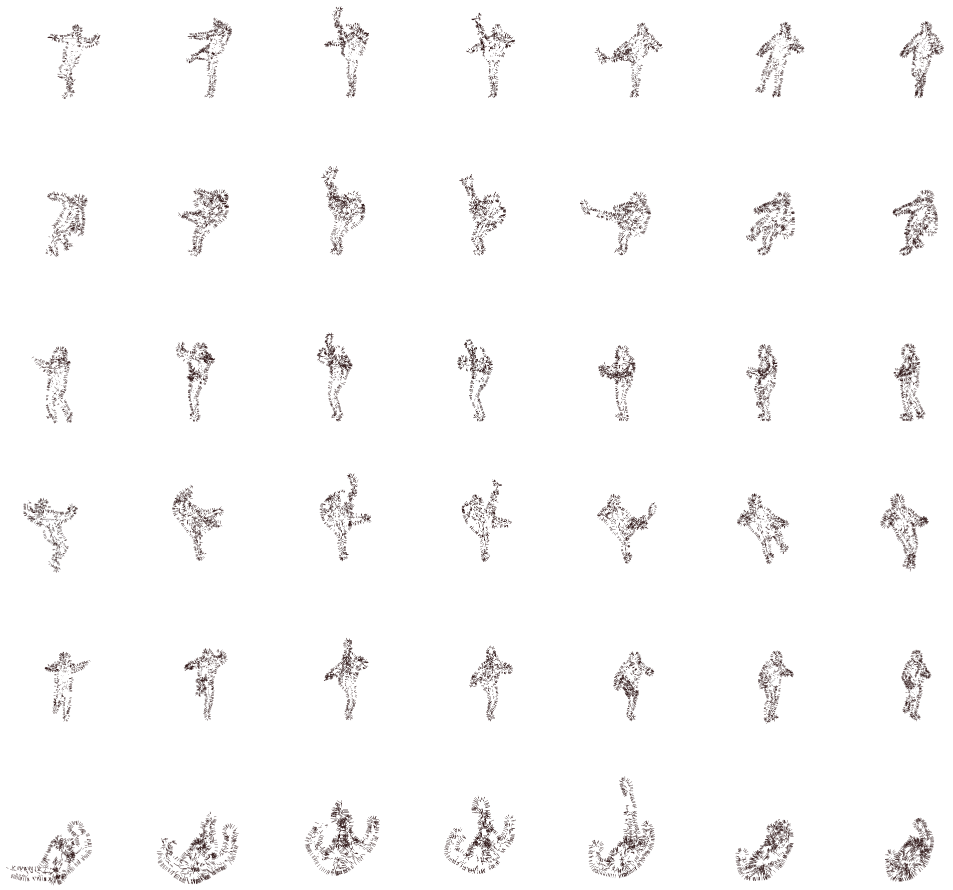


Fig. 5.3 – Les observations pour 7 trames : les points 3D et les normales à la surface, montrées suivant les 6 points de vue des caméras utilisées.

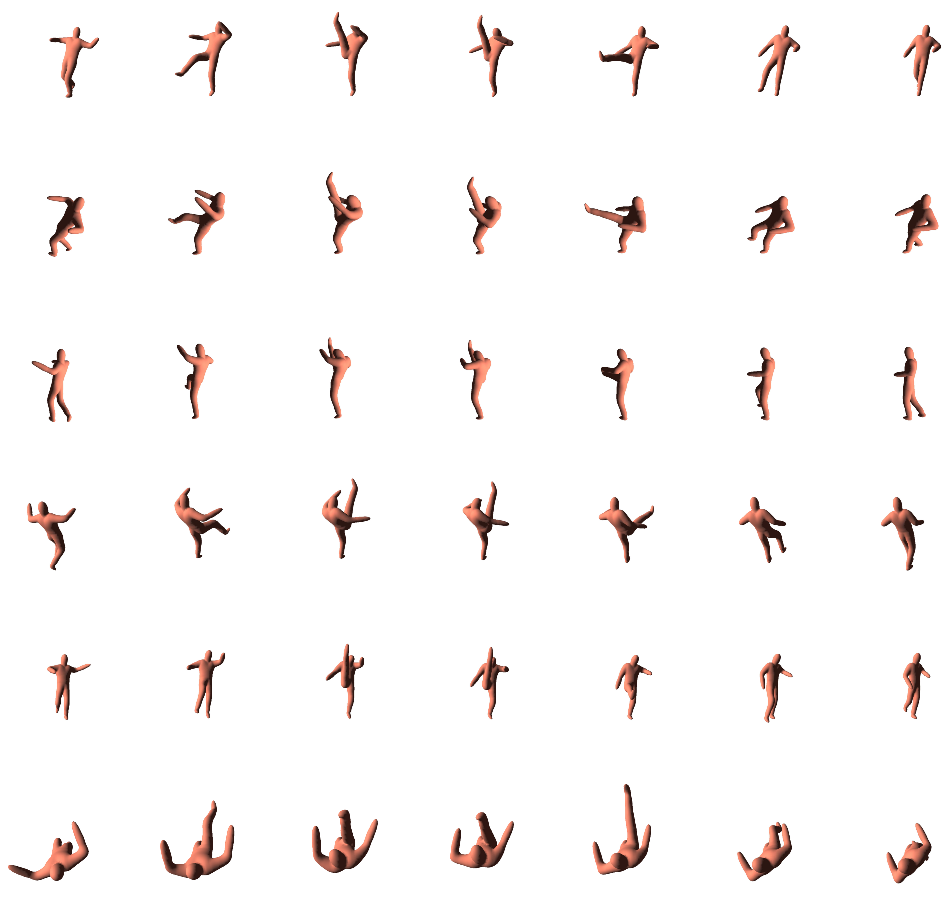


FIG. 5.4 – La pose du modèle déterminée pour 7 trames et suivant les 6 points de vue des caméras utilisées.

correspondance avec les observations, des points sur l'axe médian, supporte une certaine imprécision dans les longueurs des segments. Les segments sont reliés par des joints en rotation qui modélisent les articulations ; le modèle possédant au total 21 degrés de liberté en rotation et 3 en translation/position.

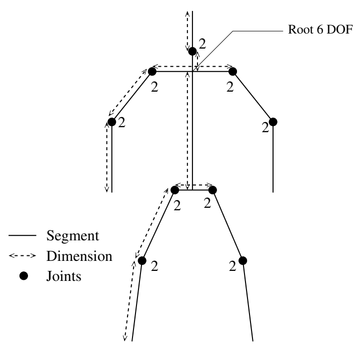


FIG. 5.5 – Le modèle avec 24 degrés de liberté pour la posture.

5.2.2 Les données

Les données utilisées sont issues ici aussi du processus de modélisation à partir de plusieurs silhouettes. Les silhouettes permettent de calculer une forme visuelle de l'objet observé, l'enveloppe visuelle ou une autre forme visuelle. Nous utilisons l'*axe médian* de cette forme comme donnée d'observation du squelette. L'axe médian d'une surface fermée est défini comme le lieu des centres des boules qui sont maximales à l'intérieur de la surface[Ser 82]. Dans le cas discret, nous calculons une approximation de l'axe médian constituée de sommets du diagramme de Voronoi de la forme calculée. Les points tridimensionnels correspondants constituent des points de mesure auxquels le squelette est mis en correspondance. La figure 5.6 montre 7 trames suivant les 6 points de vue des caméras utilisées. Il est à noter ici que l'axe médian d'une surface fermée peut lui-même être une surface dans l'espace, alors que le squelette est une courbe. Néanmoins, cela ne constitue pas un obstacle majeur car seul le torse est réellement concerné par ce problème lors de la mise en correspondance, les autres parties du corps, les bras et les jambes par exemple, produisent en effet des observations qui décrivent des courbes. Et l'expérience montre que la mise en correspondance par les moindres carrés positionne assez naturellement le squelette au milieu du torse.

5.2.3 La mise en correspondance

La mise en correspondance entre les observations et le modèle s'effectue ici aussi en minimisant les distances entre les points de l'axe médian et le squelette. Ces distances ne sont pas calculées entre un point et toutes les parties du squelette mais entre un point et une partie du squelette identifiée par une procédure EM[2]. La figure 5.7 illustre les résultats obtenus avec cette méthode sur 7 trames

[2] *Expectation Maximisation* [Dem 77] est une méthode standard pour estimer des paramètres cachés, ici l'association entre chaque points de l'axe médian et les différentes parties

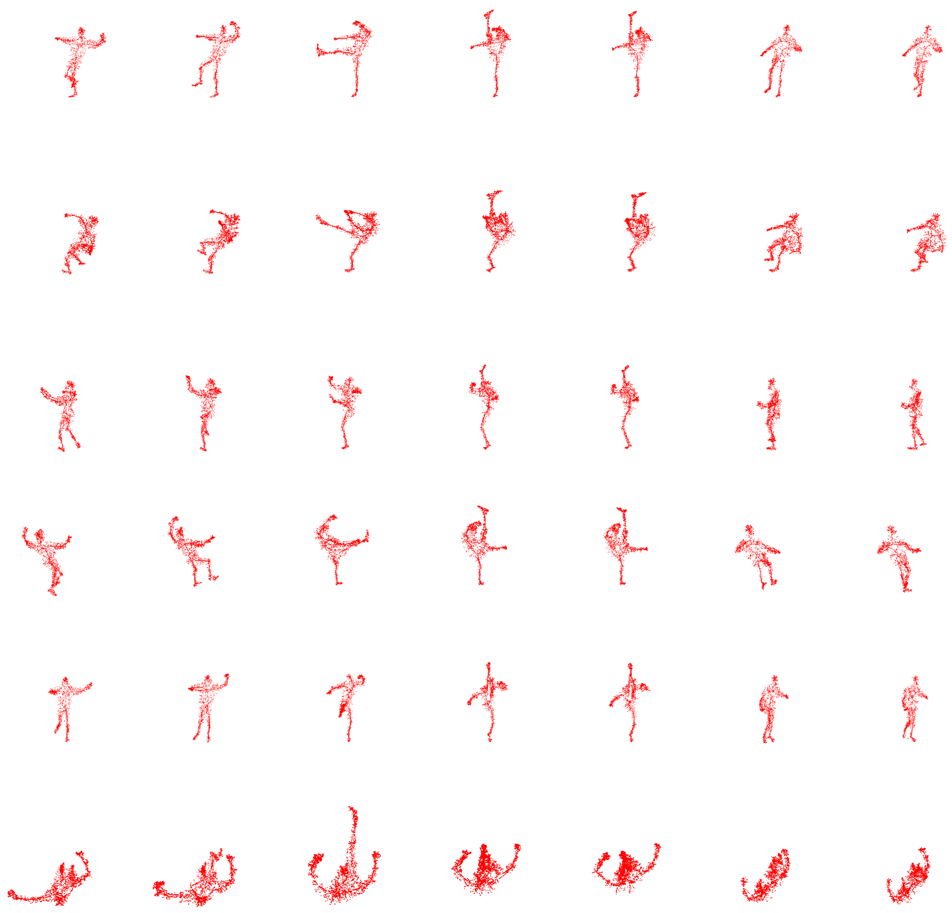


FIG. 5.6 – Les observations pour 7 trames : les points de l'axe médian de l'enveloppe visuelle suivant les 6 points de vue des caméras utilisées.

et suivant les 6 points de vue des caméras utilisées.

du squelette

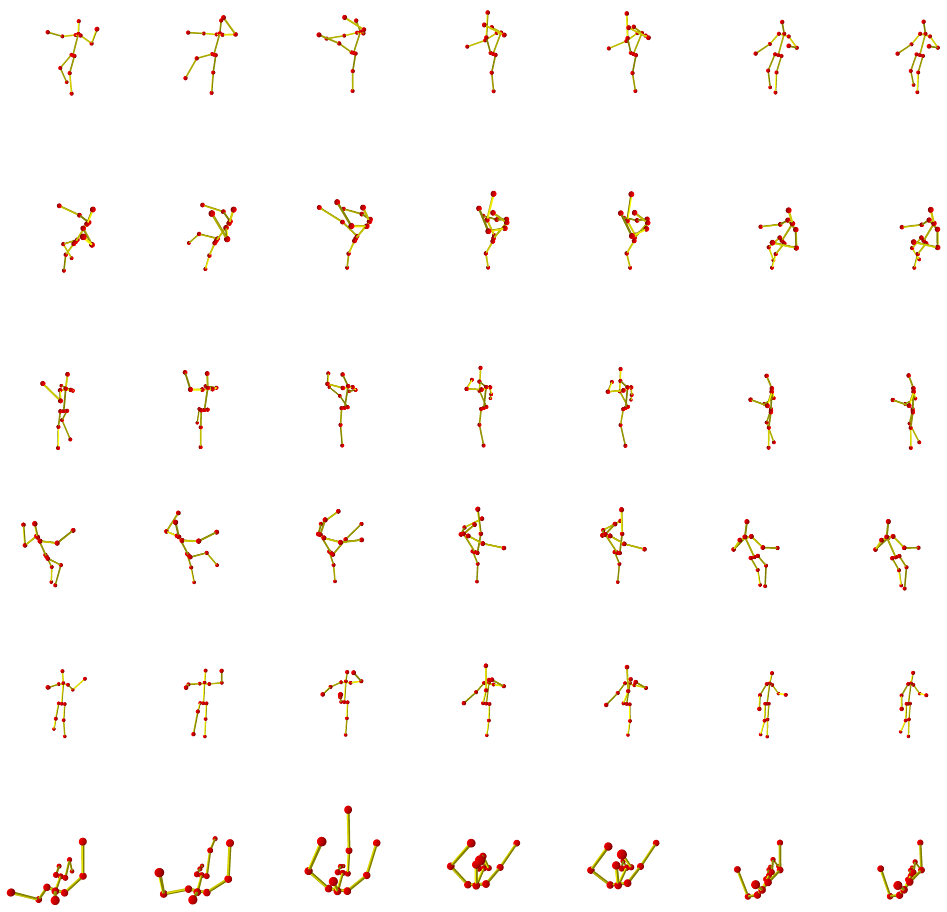


FIG. 5.7 – La pose du modèle déterminée pour 7 trames et suivant les 6 points de vue des caméras utilisées.

Chapitre 6

Articles

6.1 On Computing Exact Visual Hulls of Solids Bounded by Smooth Surfaces

Svetlana Lazebnik, Edmond Boyer and Jean Ponce

Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition

Vol. I, pp. 156-161, Kauai (USA), December 2001

On Computing Exact Visual Hulls of Solids Bounded by Smooth Surfaces

Svetlana Lazebnik
Beckman Institute
University of Illinois, Urbana, USA
slazebni@uiuc.edu

Edmond Boyer
Movi–Gravir–Inria Rhône-Alpes
Montbonnot, France
Edmond.Boyer@inrialpes.fr

Jean Ponce
Beckman Institute
University of Illinois, Urbana, USA
ponce@cs.uiuc.edu

Abstract

This paper presents a method for computing the visual hull that is based on two novel representations: the rim mesh, which describes the connectivity of contour generators on the object surface; and the visual hull mesh, which describes the exact structure of the surface of the solid formed by intersecting a finite number of visual cones. We describe the topological features of these meshes and show how they can be identified in the image using epipolar constraints. These constraints are used to derive an image-based practical reconstruction algorithm that works with weakly calibrated cameras. Experiments on synthetic and real data validate the proposed approach.

1. Introduction

Most algorithms for surface reconstruction from outlines compute some form of the *visual hull* [10], or the intersection of solid visual cones formed by back-projecting silhouettes found in the input images. The basic approach dates back to Baumgart's 1974 PhD thesis [1], where a polyhedral visual hull is constructed by intersecting the viewing cones associated with polygonal silhouettes. Volume intersection has remained the dominant paradigm for decades, implemented using representations as diverse as octrees [15] and triangular splines [14]. More recently, graphics researchers have presented efficient algorithms that avoid general 3D intersections by taking advantage of epipolar geometry [11, 13]. Given an image sequence from a camera undergoing a continuous motion, it is also possible to avoid explicit intersections by reconstructing the visual hull as the envelope of the surface tangent planes along the smoothly deforming occluding contours [2, 3, 16].

Defined in full generality, the visual hull is the maximal shape consistent with an object's silhouettes as seen from any viewpoint in a given region, and the *exact* visual hull is the visual hull with respect to a continuous region of space surrounding the object [10]. In this paper we do not treat such limiting cases, but consider the visual hull associated with a finite number of isolated viewpoints (in this discrete formulation, the volume intersection and the silhouette-consistency definitions are equivalent). We represent the visual hull as a generalized polyhedron: the faces on its surface are visual cone patches, edges are intersection curves between two viewing cones, and vertices are isolated points where more than two faces meet. In this context, a visual hull is exact when it correctly captures the connectivity of these features. Based on this notion of exact visual hulls, we specify a novel reconstruction algorithm that does not rely on polyhedral intersections or voxel-based carving, and produces precise topological and geometric meshes.

2. Preliminaries

We assume that we are observing a solid object with weakly calibrated pinhole cameras. The surface of the object is smooth and without planar patches, and the cameras are in general position. It is also assumed that apparent contours of the object have been identified in each input view and oriented counterclockwise, so that the image of the object always lies to the left of the contour. To simplify the presentation, we also assume that contours do not contain singularities such as T-junctions and cusps, and restrict our attention to objects of genus 0.

In the rest of the paper, we use the following terminology. The *rim* or *contour generator* associated with a camera is the set of all surface points where the optical ray through the pinhole grazes the object. For general viewpoints, the rim is a smooth space curve without singularities [4]. Two rims can intersect at isolated points on the surface, called *frontier points* [3, 8, 12], where the viewing rays from both pinholes lie in the surface tangent plane. The projection of a rim onto the image plane of a camera is the *apparent contour*. The set of rays from one camera center passing through points on the surface forms the *visual cone* associated with that camera. As described in the introduction, the solid formed by the intersection of all given viewing cones is the *visual hull*. Note that the shape of the viewing cones depends only on the camera center and on the shape of the object, not on the position of the image plane. Thus, projective geometry is sufficient to describe the structure of the visual hull. In particular, it is possible to develop a visual hull algorithm that relies only on weak calibration.

Imagine sweeping out a cone of optical rays along one apparent contour. Since each ray must graze the object along the rim, each ray must lie on the visual hull for some non-empty interval around its point of tangency with the object surface. In this way, each ray contributes an interval to the surface of the visual hull, and the collection of these intervals along all rays forms a *cone strip* that continuously bounds the rim on each side. Strips are delimited by segments of *intersection curves* between pairs of visual cones. An intersection curve generally does not lie on the surface, except at frontier points, where the tangent planes to the two cones and to the object coincide [5]. At these points, the intersection curve is singular: it has four branches that converge to create a characteristic X-shape (see Figure 1).

Figure 2 shows an example of an ovoid observed by three cameras. Three viewing cones are drawn, along with intersection curves and rims. The figure shows frontier points and *triple points* where three viewing cones intersect. In the figure, each frontier point is incident to four rim arcs and four intersection curve branches, and each triple point is incident to six intersection curve branches, only three of which belong to the visual hull. It can be shown that these incidence relations hold in general.

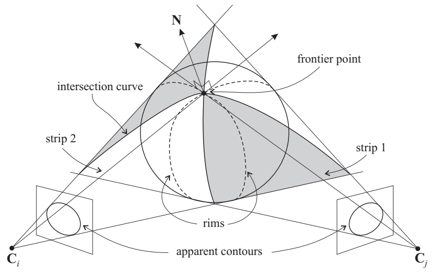


Figure 1: A surface observed by two cameras. The two rims intersect at the frontier point where two cone strips cross.

Now we introduce the two meshes computed by our algorithm. The *rim mesh* is defined on the surface of the actual object. Its vertices are frontier points, edges are rim segments between two successive frontier points, and faces are regions of the surface bounded by two or more edges. A conceptual precursor of the rim mesh is the *epipolar net* of Cross and Zisserman [5], who informally discuss, but do not construct, the arrangement of rims on the surface of an object as the camera moves. The *visual hull mesh* is a topological description of the configuration of visual cone patches on the surface of the solid formed by the intersection of all given visual cones. Its vertices are frontier points (where two strips cross) and triple points (where three cones intersect), edges are intersection curve segments between two consecutive vertices, and faces are the cone patches that make up the strips. Examples from Figures 1 and 2 illustrate the fact that successive frontier points on one rim break up the cone strip along that rim into separate faces. Thus, there exists a one-to-one relationship between edges of the rim mesh and faces of the visual hull mesh. Continuing with the example of Figure 2, Figure 3 shows the rim and visual hull meshes of the ovoid.

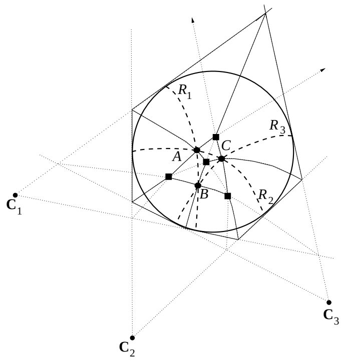


Figure 2: Configuration of rims and intersection curves for an ovoid observed by three cameras. Rims are dashed arcs, frontier points are labeled dots, and triple points are squares. Dotted arcs are intersection curve branches outside the visual hull.

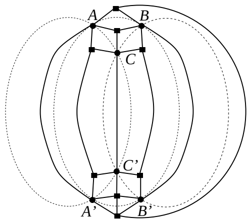


Figure 3: The rim and visual hull meshes of the ovoid from Figure 2. Frontier points are circles, and triple points are squares. Rim segments (edges of the rim mesh) are dashed. Intersection curve segments (edges of the visual hull mesh), are bold lines. Note that frontier points A', B', and C' belong to the region of the surface not visible in the previous figure.

3. Computing the Rim Mesh

We begin by computing the frontier points, which are the vertices of the rim mesh. At a frontier point $\mathbf{P}_{ij}$ due to views i and j, the tangent plane to the surface is also the epipolar plane determined by $\mathbf{P}_{ij}$ and the camera centers $\mathbf{C}_i$ and $\mathbf{C}_j$. In the images, this means that corresponding epipolar lines $\mathbf{l}_{ij}$ and $\mathbf{l}_{ji}$ are both tangent to the respective contours at the projections $\mathbf{p}_i$ and $\mathbf{p}_j$ of $\mathbf{P}_{ij}$. Figure 4 illustrates this basic setup, along with other notation that we will need later. Finding a pair of matching frontier points in images i and j is a one-dimensional search problem, where

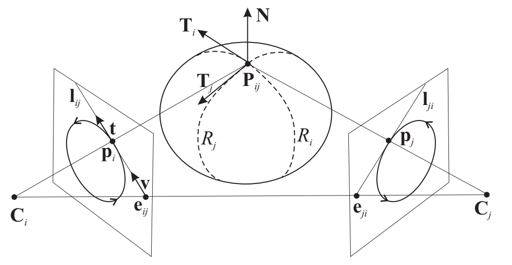


Figure 4: $\mathbf{P}_{ij}$ is a frontier point where two rims R_i and R_j intersect. $\mathbf{P}_{ij}$ projects onto points $\mathbf{p}_i$ and $\mathbf{p}_j$ at which the respective contours are tangent to the two matching epipolar lines $\mathbf{l}_{ij}$ and $\mathbf{l}_{ji}$.

we parametrize the pencil of epipolar lines by their slope and look for line $\mathbf{l}_{ij}$ that is tangent to the ith contour, such that the corresponding line $\mathbf{l}_{ji}$ in the jth image is tangent to the jth contour. In the presence of contour extraction and calibration errors, there may not exist a pair of matching epipolar lines that exactly satisfy the tangency constraint. In this situation, we find approximately matching frontier points such that the angle difference between the tangent line in one image and the reprojected epipolar tangent from the other image is minimized. Difficulties caused by data error will be further discussed in Section 4.

We obtain all frontier points by matching their projections in two images. The four rim edges incident to a particular frontier point are given by intervals on the two apparent contours that are incident to this point. Thus, there exists a one-to-one correspondence between contour segments in the images and edges of the rim mesh. The orientation of rim edges is then given by the orientation of the corresponding contour segments. In this way, we obtain the complete adjacency information for edges and vertices of the rim mesh. To compute the faces, we need to know the relative ordering in space of the four rim segments incident on each vertex. More formally, we associate with each frontier point $\mathbf{P}_{ij}$ a circular list I where the four edges appear in CCW order around the surface normal. Let $\mathbf{T}_i$, $\mathbf{T}_j$ be the tangents to the rims R_i and R_j at $\mathbf{P}_{ij}$ (see Figure 4). Then the index i appears before the index j in the ordered list I iff

$$(\mathbf{T}_i \wedge \mathbf{T}_j) \cdot \mathbf{N} > 0, \qquad (1)$$

where $\mathbf{N}$ is the outward-pointing surface normal (computed as the cross product of the oriented tangent to the contour and the viewing direction).

Equation (1) gives rim ordering in terms of of tangents to the apparent rims, which cannot be computed given only image information. Therefore, we need an equivalent image-based expression for rim ordering. Consider image i and let $\mathbf{v}$ be the direction from the epipole $\mathbf{e}_{ij}$ to $\mathbf{p}_i$, the projection of $\mathbf{P}_{ij}$. Let also $\mathbf{t}$ be the tangent to the contour at $\mathbf{p}_i$ (see Figure 4), and let k_s be the apparent curvature at $\mathbf{p}_i$. Then rims R_i and R_j are CCW oriented at $\mathbf{P}_{ij}$ iff

$$(\mathbf{v} \cdot \mathbf{t})\ k_s > 0. \qquad (2)$$

The above expression is equivalent to (1), and the following is a brief sketch of the proof. When the viewing direction rotates in the surface tangent plane around the normal at $\mathbf{P}_{ij}$, the tangent to the rim also rotates. The directions of camera and tangent rotation are the same if the surface is elliptic at $\mathbf{P}_{ij}$ and opposite if the surface is hyperbolic. This is a direct consequence of the fact that the Gauss map of the surface is orientation-preserving at elliptic points and orientation-reversing at hyperbolic points [6]. Thus two rims R_i and R_j are CCW oriented if (a) the camera rotates CCW around the normal from position i to j and the surface is elliptic at $\mathbf{P}_{ij}$; or (b) the camera rotates CW and the surface is hyperbolic. The sign of $\mathbf{v} \cdot \mathbf{t}$ is positive if the camera rotates CCW in the tangent plane and negative otherwise. Moreover, the sign of the apparent curvature k_s is positive if the surface is elliptic or negative if the surface is hyperbolic [9]. Thus, expression (2) is positive in the two above mentioned cases and negative otherwise. It is therefore the desired image-based expression equivalent to (1).

Given the above rim ordering criterion, it is straightforward to trace the loops of edges bounding rim faces. Suppose we start with one rim segment s of the rim R_i and want to find the face that lies to its left. Informally, we traverse s along its direction and at its endpoint $\mathbf{P}_{ij}$, simply take a left turn to get to the next edge, that is, we select the edge preceding s in the ordered circular list of $\mathbf{P}_{ij}$. We move from endpoint to endpoint in this manner, traversing edges either forward or backward along their orientation, taking a left turn each time until we complete a cycle.

4. Computing the Visual Hull Mesh

As demonstrated in the previous section, we can compute the topology of the rim mesh without knowing anything about its geometry, except for the positions of frontier points (note that under weak calibration we can only recover these positions up to a projective transformation). This topology constrains the adjacency relationships between triple points and intersection curves, which are the vertices and edges of the visual hull mesh.

A triple point $\mathbf{P}_{ijk}$ the intersection of three optical rays back-projected from contour points $\mathbf{p}_i$, $\mathbf{p}_j$, and $\mathbf{p}_k$ in three different images (see Figure 5). In particular, $\mathbf{p}_k$ satisfies the *transfer equation* [7]

$$\mathbf{p}_k = \mathbf{l}_{ki} \wedge \mathbf{l}_{kj} = (\mathbf{F}_{ik}\,\mathbf{p}_i) \wedge (\mathbf{F}_{jk}\,\mathbf{p}_j), \qquad (3)$$

where $\mathbf{F}_{mn}$ is the *fundamental matrix* mapping points in image m to epipolar lines in image n, and homogeneous

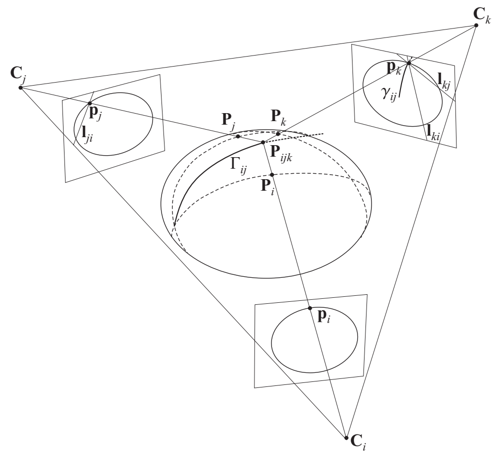


Figure 5: The triple point $\mathbf{P}_{ijk}$ is a "phantom point" where the rays formed by back-projecting epipolar correspondents $\mathbf{p}_i$, $\mathbf{p}_j$, and $\mathbf{p}_k$ meet in space. $\mathbf{P}_{ijk}$ can be located by tracing the intersection curve Γ_{ij} between views i and j and noticing when its projection γ_{ij} in the kth image crosses the contour.

coordinates are used for image points. Points $\mathbf{p}_i$ and $\mathbf{p}_j$ satisfy symmetric equations. Any pair of $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{p}_k$ are *epipolar correspondents* — that is, any two of the points lie in the epipolar plane defined by the two camera centers and one of the points [2]. A triple point is a standard trinocular stereo correspondence, but it does not usually lie on the surface because $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{p}_k$ are projections of three different points $\mathbf{P}_i$, $\mathbf{P}_j$, and $\mathbf{P}_k$ on three different rims.

Just as with finding frontier points, finding triple points is a one-parameter search. We walk along the ith contour in discrete steps and for each contour point $\mathbf{p}_i$ find the epipolar line $\mathbf{l}_{ji} = \mathbf{F}_{ij}\,\mathbf{p}_i$ in image j, and locate an epipolar correspondent $\mathbf{p}_j$ by intersecting $\mathbf{l}_{ji}$ with the jth contour. We then obtain a third point $\mathbf{p}_k$ by transferring $\mathbf{p}_i$ and $\mathbf{p}_j$ using (3) and check whether $\mathbf{p}_k$ lies on the kth contour. As shown in Figure 5, transfer of successive epipolar correspondents allows us to trace the intersection curve Γ_{ij} between ith and jth cones in the kth image. The triple point is revealed when the traced curve crosses the kth contour.

In general, epipolar correspondents are not unique. In the case shown in Figure 5, each epipolar line intersects each contour twice (this reflects the fact that intersection curves have multiple branches). Moreover, the epipolar correspondence criterion does not say when a triple point belongs to the visual hull — the additional constraint is that the point must not project outside the silhouette in any other input view. However, if we are able to exactly compute the positions of frontier points along the contours, we can use this information to simplify the search for triple points.

Each face of the rim mesh is bounded by rim segments that also belong to the surface of the visual hull, so we can identify faces with regions on the surface of the visual hull that have the same boundary. Consider a single face f of the rim mesh. The edges of f tell us which viewing cones contribute to the visual hull inside the region identified with f, and give us a corresponding subset of contours that need to be searched for triple points belonging to this region. Since each edge of f corresponds to a single contour interval in some image, any triple point that belongs to f must project to a point along these intervals. Thus, it is sufficient to trace intersection curve segments between each pair of these intervals and find triple points when the curve being traced goes outside the contour in one of the other views that contribute to f. Taking each face separately, we compute triple points, intersection curves, and their connectivity. Since we know which pair of cones gave rise to each segment of an intersection curve, we can identify all the segments bounding a cone strip. Individual faces of the strips are identified by grouping all the intersection curve segments that project within the contour interval that corresponds to a particular rim segment.

The algorithm described above yields the correct visual hull mesh given exact input data (perfectly extracted contours, error-free fundamental matrices). However, noise and calibration error tend to destroy exact topological features, most importantly, intersection curve crossings at frontier points. Figure 6 illustrates the situation. The epipolar line $\mathbf{l}_{ij}$ is tangent to the contour at point $\mathbf{p}_i$ in the ith image. This line corresponds to the line $\mathbf{l}_{ji}$ in the jth image, which is not tangent to the jth contour, but intersects it in two epipolar correspondents $\mathbf{p}_{j1}$ and $\mathbf{p}_{j2}$. Intersecting the visual rays due to these three points in the epipolar plane yields two distinct intersection curve points $\mathbf{P}_{ij1}$ and $\mathbf{P}_{ij2}$, instead of a single frontier point. Thus, instead of being singular, the intersection curve separates into two distinct branches that do not meet. In order to approximate the position of a frontier point, we have to match $\mathbf{p}_i$ with the epipolar tangency point $\mathbf{p}_j$ in the jth image. However, the two points do not lie in the same epipolar plane, and visual rays through them do not intersect. We could estimate the location of $\mathbf{P}_{ij}$ as the midpoint of the segment connecting the points of closest approach of the two rays, but this approximated point does not lie on the traced intersection curves. This leads to serious consistency problems for a naive implementation that attempts to strictly enforce combinatorial constraints on exact visual hull structure.

Intuitively, small perturbations to exact contour and calibration data result in contours that back-project to general cones in space, the intersection of which does not have to share the properties of exact visual hulls. For instance, while we know that cone strips never break up in theory (each ray interval along the strip must contain at least one point), in noisy data, they may break up near the frontier points as shown in Figure 6. Nevertheless, even with large

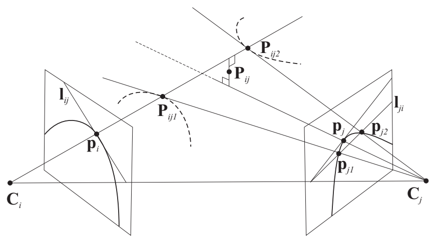


Figure 6: Tracing intersection curves given inexact data (see text). Intersection curve branches are shown as dashed lines.

errors in the data, the intersection of cones in space is still well defined, and we can compute it using a variant of our exact algorithm. We trace entire intersection curves, as opposed to breaking them up into pieces belonging to separate faces of the rim mesh, and then clip out all components of the curves that project outside any of the silhouettes. In the process, combinatorial information about the curves is maintained, so that it becomes possible to recover the geometry of cone strips. Namely, boundary points of the strips are connected in the order induced by the contour parametrization, and are separated into two groups that correspond to segments bounding the rim on the near and the far side with respect to their distance from the camera. This data structure is a monotone polygon, and it can be triangulated in linear time for purposes of display.

5. Experimental Results

The first input sequence consists of six synthetically generated images of an egg model. Contours were extracted using snakes and modeled as cubic B-splines. As seen in Figure 7, the algorithm correctly generates the rim mesh and the visual hull, both in their topological and geometric form. The contrast between these two forms is clearly visible by comparing two renderings of the same strip in Figure 7 (f) and (g). The exact strip explicitly shows frontier points and triple points, and intersection curves are forced to converge in four branches at frontier points. The triangulated strip does not degenerate at frontier points, because the robust strip tracing algorithm ignores them.

We also demonstrate results for two calibrated nine-image turntable sequences of a gourd and a vase (Figure 8). For both of these data sets, the algorithm constructs a complete rim mesh, even though many of the frontier points are densely clustered near the top and the bottom. To better visualize the structure of these meshes, we rendered their vertices and edges as graphs using a publicly available graph drawing program. The graphs, shown in Figure 8 (b) and (h), reveal the regular structure of rim crossings which is impossible to observe in the images themselves. Each rim mesh in our examples obeys Euler's formula for topological polyhedra of genus 0, $V + F = E + 2$ (note that the meshes are actually planar, even though the graph layouts shown are not). Because of stability problems inherent in real-world data, the algorithm does not recover topological visual hull meshes for these two data sets. Instead, we obtain precise geometric models of the visual hulls that do not capture every triple point, but are suitable as input for common modeling and rendering applications. Figure 8 shows these models, along with selected strips. Note that the strips degenerate completely for relatively large intervals near the top and the bottom, in the areas of dense frontier points. This behavior is not possible in theory, but it occurs in practice, as discussed in Section 4.

6. Discussion and Future Work

Our preliminary results are intriguing. Significantly, the recovery of exact rim meshes has proven to be robust even with densely clustered frontier points that do not lie on matching epipolar lines. Note that the rim mesh structure depends only on the relative ordering of frontier points along the rims, not on absolute positions — hence the relative stability of the topology. With visual hull meshes, the situation is different: the connectivity of intersection curves and triple points is elusive, while the geometry may still be recovered reliably. It will be important to investigate the question of whether these instabilities are inherent in the conditioning of exact visual hull computation, or are introduced by our algorithm. We are considering a different approach to recovering the topology of the visual hull mesh that would take full advantage of the combinatorial constraints given by the structure of the rim mesh. We are also working on extending our implementation to deal with T-junctions and surfaces of arbitrary genus, to handle more complex and visually interesting objects.

Overall, our approach has several attractive features. Most importantly, it is based on an analysis of the exact structure of visual hulls from finitely many viewpoints, which has received little attention in previous research. Our approach takes advantage of epipolar geometry and weak calibration — in a sense, we don't need to know where the cameras are. Moreover, our algorithm uses only two-dimensional computations, and constructs a range of shape representations, from graph-theoretic and topological, to completely image-based, to purely geometric. For these reasons, it brings fresh insights to the theory and practice of the venerable problem of visual hull computation.

Acknowledgments. This work was supported in part by the Beckman Institute, the National Science Foundation under grant IRI-990709 and a UIUC-CNRS collaboration agreement.

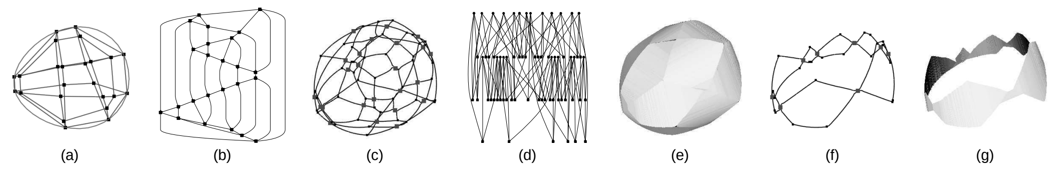


Figure 7: Egg results. (a) the rim mesh superimposed on one of the input outlines; (b) the vertices and edges of the rim mesh shown as a graph (24 frontier points, 48 edges, 26 faces); (c) the visual hull mesh from shown from a viewpoint not in the input set; (d) a graph of the visual hull vertices and edges (68 vertices, 114 edges, 48 faces); (e) a triangulated geometric model of the mesh; (f) one of the strips making up the exact visual hull mesh; (g) a triangle model of the same strip.

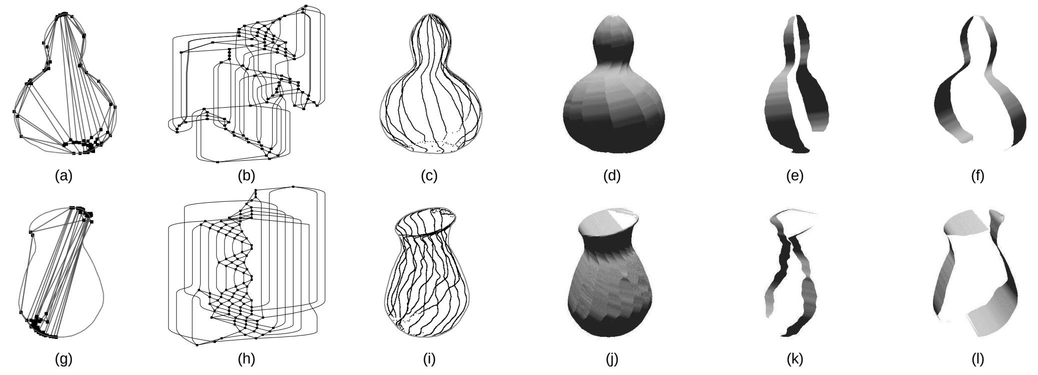


Figure 8: Top row: gourd results. (a), (b) the rim mesh and corresponding graph (96 vertices, 192 edges, 98 faces); (c) intersection curves that make up the visual hull; (d) triangulated visual hull model; (e), (f) two of the strips from the model. Bottom row: teapot results. (g), (h) rim mesh (104 vertices, 208 edges, 106 faces); (i) intersection curves on the surface of the visual hull; (j) visual hull model; (k), (l) two cone strips.

References

[1] B.G. Baumgart, "Geometric Modeling for Computer Vision", Ph. D. Thesis (Tech. Report AIM-249), Stanford University, 1974.

[2] E. Boyer and M. Berger, "3D Surface Reconstruction Using Occluding Contours", *Int. J. Comp. Vision*, 22(3), 1997, pp. 219-233.

[3] R. Cipolla, K. Astrom, and P.J. Giblin, "Motion from the Frontier of Curved Surfaces", *Proc. IEEE Int. Conf. on Comp. Vision*, 1995, pp. 269-275.

[4] R. Cipolla and P.J. Giblin, *Visual Motion of Curves and Surfaces*, Cambridge University Press, Cambridge, 1999.

[5] G. Cross and A. Zisserman, "Surface Reconstruction from Multiple Views Using Apparent Contours and Surface Texture", *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics*, 2000, pp. 25-47.

[6] M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[7] O. Faugeras and L. Robert, "What Can Two Images Tell Us About a Third One?", *Int. J. Comp. Vision*, 18(1), 1996, pp. 5-19.

[8] P.J. Giblin and R. Weiss, "Epipolar Curves on Surfaces", *Image and Vision Computing*, 13(1): pp. 33-44, 1995.

[9] J. Koenderink, "What Does the Occluding Contour Tell Us About Solid Shape?", *Perception*, 1984, pp. 321-330.

[10] A. Laurentini, "The Visual Hull Concept for Silhouette-based Image Understanding", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(2), 1994, pp. 150-162.

[11] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based Visual Hulls", *Proc. SIGGRAPH* 2000, pp. 369-374.

[12] J. Porrill and S. Pollard, "Curve Matching and Stereo Calibration", *Image and Vision Computing*, 9(1): pp. 45-50, 1991.

[13] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller, "Reconstructing 3D Tree Models from Intrumented Photographs", *IEEE Computer Graphics and Applications*, 21(3), 2001, pp. 53-61.

[14] S. Sullivan and J. Ponce, "Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs using Triangular Splines", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(10), 1998, pp. 1091-1096.

[15] R. Szeliski, "Rapid Octree Construction From Image Sequences," *CVGIP: Image Understanding*, 1(58), 1993, pp. 23-32.

[16] R. Vaillant and O. Faugeras, "Using Extremal Boundaries for 3-D Object Modeling", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2), 1992, pp. 157-173.

6.2 A Hybrid Approach for Computing Visual Hulls of Complex Objects

Edmond Boyer and Jean-Sébastien Franco

Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition

Vol. I, pp. 695-701, Madison (USA), June 2003

A Hybrid Approach for Computing Visual Hulls of Complex Objects

Edmond Boyer and Jean-Sébastien Franco
GRAVIR–INRIA Rhône-Alpes
655, Avenue de l'Europe, 38330 Montbonnot, France
Edmond.Boyer@inrialpes.fr, Jean-Sebastien.Franco@inrialpes.fr

Abstract

This paper addresses the problem of computing visual hulls from image contours. We propose a new hybrid approach which overcomes the precision-complexity trade-off inherent to voxel based approaches by taking advantage of surface based approaches. To this aim, we introduce a space discretization which does not rely on a regular grid, where most cells are ineffective, but rather on an irregular grid where sample points lie on the surface of the visual hull. Such a grid is composed of tetrahedral cells obtained by applying a Delaunay triangulation on the sample points. These cells are carved afterward according to image silhouette information. The proposed approach keeps the robustness of volumetric approaches while drastically improving their precision and reducing their time and space complexities. It thus allows modeling of objects with complex geometry, and it also makes real time feasible for precise models. Preliminary results with synthetic and real data are presented.

1. Introduction

Assume we are given several silhouettes of an object corresponding to different camera viewpoints. The visual hull is the maximal solid shape consistent with the object silhouettes. Such an approximation of the object captures all the geometric information available from the object silhouettes. The interest arises, therefore, in all modeling applications making use of silhouettes. In this paper we describe how to efficiently use the silhouette information to compute visual hulls. The motivation is to propose a new practical solution for computing precise models of complex objects, along with a reasonable complexity in time and space.

Visual hulls were first introduced by Laurentini [13] in the theoretical context where an infinite number of viewpoints, surrounding the object's convex hull, are considered. Before and after this work, visual hulls have also been, implicitly and explicitly, widely studied in the computer vision and computer graphics communities. In particular, it has been shown recently [14] that the visual hull of a smooth object is a topological polyhedron that can be determined using weak calibration only (oriented epipolar geometry). However, the solution given in this work does not apply to most real situations. There are many other algorithms for computing approximations of the visual hull in both communities: some consider the volume enclosed by the visual hull and are based on space discretizations; some others focus on the surface of the visual hull and consider individual points or polyhedral representations.

Volumetric approaches are based on space discretizations into elementary cells, the voxels, which are carved according to their image positions with respect to the silhouettes. An early approach was proposed by Martin and Aggarwal [15] who used parallelepipedic cells aligned with the coordinate axis. Later on, octrees were proposed [5] as adaptive data structures for representing visual hulls and efficient approaches [21, 18, 4] were presented to compute voxel-based representations. These approaches are purely geometric and do not consider photometric information. Recent methods [12] make use of such information and carve voxels according to the *color consistency* of their projections onto the different images. See [19] and [9] for reviews on volumetric approaches for modeling. All the aforementioned approaches are based on regular voxel grids and can handle objects with complex geometries. However, the 3D space discretizations used are computationally expensive and lack precision since most of the grid points do not belong to the visual hull surface under consideration.

Surface-based approaches use a different strategy. Visual hull boundary elements, points and faces, are estimated by intersecting the viewing cone surfaces associated with the occluding contours. Baumgart [2] made an early contribution using polygonal approximations of the occluding contours. [11, 6, 3] focused on individual points reconstructed using local second order surface approximations. More recently, approaches have been proposed to compute surface patches[20], or strips [16] of the visual hull. Surface-based approaches can be precise, especially compared to volumetric approaches, however the surface models produced are often incomplete or corrupted, in particular when considering complex objects. A reason for this is the fact that intersections of viewing cone boundaries are generally not well defined, and thus very sensitive to numerical instabilities.

Related to surface-based approaches, Matusik *et al.* [17] have shown that 2D calculations are sufficient when computing new images of an object using its visual hull. This interesting result follows the fact mentioned earlier that the visual hull is a projective structure [14], the approach, however, does not lead to geometric models as required in many applications.

Our approach takes advantages of both categories described above. It uses the robustness of volumetric approaches while keeping the precision of surface-based approaches. A space discretization into cells is still used, but unlike most volumetric algorithms, sample points are not regularly spaced but computed on the surface of the visual hull. Elementary cells are then tetrahedrons coming from the Delaunay triangulation applied on the sample points. The final polyhedral model is obtained by carving these cells according to their image projections. This approach presents two important contributions with respect to the methods mentioned previously: first, the points used to construct the model lie on the surface of the visual hull, thus enabling a high level of precision; second the surface representation is obtained by means of the Delaunay triangulation, hence ensuring robustness, and for which fast implementations exist.

The paper is organized as follows. Section 2 introduces definitions which are used in the paper. Section 3 describes how points on the visual hull are computed. Section 4 details the polyhedral representation algorithm. Experimental results are presented in section 5, before concluding with potential extensions of this work.

2. Definitions

Contours We assume that a scene, composed of several objects, is observed by a set of pinhole cameras. The objects' surfaces are supposed to be orientable closed surfaces, smooth or polyhedral. No assumption is made on their genus which may be non-zero. *Rims* are the locus of points, on the object surface, where viewing rays are tangent to the surface. Rims project onto image curves, called the *occluding contours* [15], which border the object silhouettes in the image plane. In what follows, subscripts will denote contour numbers and superscripts image numbers, thus O^i_j denotes the jth occluding contour in image i. Occluding contours are oriented in the images. Their orientation is such that the object is on the left of the oriented contour. Hence, exterior contours are oriented counterclockwise and interior contours are oriented clockwise. We will call the *inside region* of an occluding contour the closed region of the image plane delimited by the contour and containing the silhouette, and we will call the *outside region* its complement in the image plane (see figure 1).

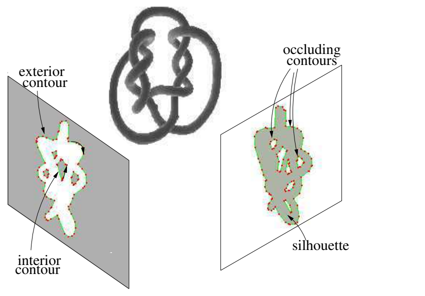


Figure 1: The occluding contours delimit the object silhouette in the image plane. The shaded region on the left image represents image points which are outside at least one contour. Its complement, shaded in the right image, represents image points which are inside all the contours, and thus belong to the silhouette.

Viewing cones Intuitively, a viewing cone is a generalized cone whose apex is the image center and whose base is the inside region of an occluding contour. More formally, the *viewing cone* $\mathcal{V}^i_j$ associated with the occluding contour O^i_j is the closure of the set of rays passing through points inside O^i_j and through the camera center of image i. $\mathcal{V}^i_j$ is thus tangent to the corresponding object surface along the rim that projects onto O^i_j. According to the orientation of O^i_j, exterior or interior, the viewing cone $\mathcal{V}$ is an acute or obtuse cone of $\mathbb{R}^3$ respectively. Viewing cone boundaries intersect along space curves which do not lie on the surface, except at *frontier points* where rims intersect. Note that in the case of polyhedral surfaces, frontier points are not necessarily isolated and can form frontier edges.

Visual hulls The visual hull is usually defined as the intersection of all the viewing cones available from the different viewpoints, it is thus the closed space region where points project inside all the occluding contours. Let $\mathcal{I}, \mathcal{C}$ be respectively the image set and the contour set under consideration, then:

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

where $\mathcal{V}^i_j$ is the viewing cone of rim j in image i. When a finite set $\mathcal{I}$ of images is considered, the visual hull is a topological polyhedron composed of cone patches delimited by cone intersection curves [14]. In practical situations, occluding contours are approximated by 2D polygonal curves, thus viewing cones are polyhedral cones and visual hulls polyhedrons. The definition above holds if a single object is observed in every image of $\mathcal{I}$. But it can not correctly handle scenes composed of several unconnected objects, some of which may not appear in all images. To this aim, we could straightforwardly extend the definition to the union of individual visual hulls, each associated to a unique real object. Let $\mathcal{K}$ be the real object set and $\mathcal{C}_k$ be the contour set for the object k, then:

$$\begin{aligned}\mathcal{VH}(\mathcal{I},\mathcal{K}) &= \bigcup_{k\in\mathcal{K}} \mathcal{VH}(\mathcal{I},\mathcal{C}_k),\\ , &= \bigcup_{k\in\mathcal{K}} (\bigcap_{i\in\mathcal{I}_k, j\in\mathcal{C}_k} \mathcal{V}^i_j),\end{aligned} \tag{1}$$

where $\mathcal{I}_k$ is the image subset of $\mathcal{I}$ where object k appears or: $\mathcal{I}_k = \{i \in \mathcal{I} : \mathcal{O}^i_j \neq \emptyset \text{ for some } j \in \mathcal{C}_k\}$. A direct application of this definition requires the sets $\mathcal{C}_k$ and $\mathcal{I}_k$ to be known. In other words, the occluding contours of objects need to be identified over the whole image set. This operation is not necessarily easy, in particular from one image to another. Furthermore, silhouettes might overlap in one image, making the object identification difficult.

Another solution is to define the visual hull as the set of points in $\mathbb{R}^3$ that project inside one silhouette in every image where the points are visible. A first step in that direction is to consider the following expression:

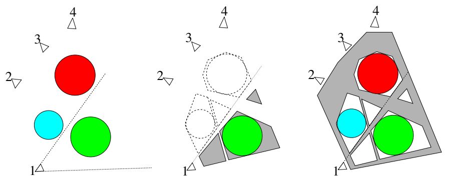
$$\mathcal{VH}(\mathcal{I},\mathcal{K}) = \bigcap_{i\in\mathcal{I}} (\bigcup_{k\in\mathcal{S}^i} (\bigcap_{j\in\mathcal{C}^i_k} \mathcal{V}^i_j)), \tag{2}$$

where $\mathcal{S}^i$ is the set of silhouettes in image i and $\mathcal{C}^i_k$ is the set of contours associated to the silhouette k in i. This expression is equivalent to (1) applied to a set $\mathcal{K}$ of virtual objects having their silhouettes either disjoint or entirely included in one another in every images. The interest is that objects'x contributions are, in that case, distinguished by their silhouettes. Since any silhouette includes exactly one exterior contour and possibly several interior contours, expression (2) can easily be applied using the exterior contours in the image set.

Nonetheless, expression (2) is not completely satisfying in its current form since it does not take into account the fact that virtual objects may not be seen in one or several images (i.e. $\bigcap_{j\in\mathcal{C}^i_k} \mathcal{V}^i_j = \emptyset$ for some k and some i) . As a consequence, they will not be part of the visual hull because they do not appear in one image contribution (see figure 2-(b)). This is due to the fact that the intersection of the image contributions in (2) should be carried out over their common domains. A simpler approach is to consider the complement of the visual hull. It is the the open region $\mathcal{VH}^c$ of $\mathbb{R}^3$ defined by:

$$\mathcal{VH}^c(\mathcal{I},\mathcal{K}) = \bigcup_{i\in\mathcal{I}} (\bigcap_{k\in\mathcal{S}^i} (\bigcup_{j\in\mathcal{C}^i_k} \mathcal{D}^i \setminus \mathcal{V}^i_j)), \tag{3}$$

where $\mathcal{D}^i$ is the image i visibility domain in $\mathbb{R}^3$ and $\mathcal{D}^i \setminus \mathcal{V}$ is the complement of $\mathcal{V}$ relative to this domain. Using (3), objects which do not appear in one image can still contribute to the visual hull since empty contributions do not affect image contributions in the above expression. Considering the visual hull or its complement is equivalent since the surface of interest borders both regions, and identifying the cells which belong to the visual hull or to its complement are dual operations. The above expression is in fact the definition that is implicitly used by volumetric approaches when carving voxels. It should be noted that expression (2) could also be modified to account for objects which are not always visible, however using the complement of the visual hull instead of the visual hull itself simplifies both expressions and algorithms.



(a) (b) (c)

Figure 2: Cross sections of a 4-viewpoints situation: (a) the original scene where camera 1 sees only the green object; (b) expression (2) is used, the visual hull (shaded) does not contain any contribution relative to the blue and red objects; (c) expression (3) is used, the complement of the visual hull (shaded) is computed and includes contributions from the blue and red objects.

Both definitions (2) and (3) may add independent virtual objects that do not appear in the original scene (as shown in figure 2). Notice however that the second definition may add more virtual objects, in particular near or far from projection centers. This is a consequence of the visibility domain constraint which limits the domain of the visual hull complement. The number and sizes of these undesired objects are usually reduced by increasing the number of viewpoints. Another solution, as implicitely used by volumetric approaches, is to use a region of interest instead of $\mathbb{R}^3$.

3. Visual hull surface points

3.1 Algorithm outline

Assume that occluding contours are extracted in the image set and let us consider a polygonal contour O^i_j in image i. Points on the associated viewing cone $\mathcal{V}^i_j$ contribute to the surface of the visual hull if: (i) they project onto O^i_j in image i, (ii) they do not project inside the intersection of silhouette complements in any other images. An obvious way to compute these points is therefore to take points on O^i_j and to look at the intersection of their viewing lines with the viewing cones originating from other viewpoints. These intersections define one or several intervals on the viewing line corresponding to the contribution of this viewing line to the surface of the visual hull.

Let p_j^i be a point of O_j^i. The contribution intervals on the viewing line of p_j^i are delimited by the intersections of the viewing line with the surfaces of the concerned viewing cones. These intervals can be determined directly in 3D by intersecting lines and cones, however, and as mentioned in [17], most of the calculations can be achieved in 2D. Indeed, points delimiting the intervals on the viewing line of p_j^i are such that their projections belong to both the epipolar line and the concerned occluding contours (see figure 3). We use these principles in algorithm 1 to reconstruct points on the visual hull surface.

Algorithm 1 Visual hull surface points

```
1:  for all contours $O_j^i$ in all images: do
2:    for all images $k$ such that $k \neq i$: do
3:      for all points $p_j^i$ in $O_j^i$ : do
4:        compute the epipolar line $l$ of $p_j^i$ in image $k$,
5:        for all contours $O_l^k$ in image $k$: do
6:          compute the intersections of $l$ with $O_l^k$,
7:          update depth intervals along the viewing line
            of $p_j^i$,
8:        end for
9:      end for
10:     compute the 3D points delimiting intervals along
        the viewing line of $p_j^i$.
11:   end for
12: end for
```

3.2 Updating depth intervals along the viewing line

As explained before, intersections of the epipolar line with occluding contours are first computed. From these intersections, we can easily compute the depths of points delimiting intervals along the viewing line, which points belong to the surface of the visual hull. The question is how to combine two lists of depths from different contours or images ?

We proceed in the following way: expression (3) is used to sum up intervals contributing to the visual hull complement, over the contour and image sets. To this aim, the object contributions, or equivalently the silhouettes, must be distinguished in each image. As explained before, this can be done by means of the exterior contours since every one of them identifies a single object. We could therefore group contours, in each image, according to the exterior contour they belong to. A simpler solution takes advantage of the fact that interior contour contributions entirely belong to the visual hull complement. Thus, only the contributions from exterior contours need to be intersected when computing the whole contribution of an image. The corresponding expression for definition (3) becomes:

$$\mathcal{VH}^c(\mathcal{I},\mathcal{K}) = \bigcup_{i\in\mathcal{I}} \left[\Big(\bigcap_{j\in \mathrm{Ext}^i} \mathcal{D}^i \setminus \mathcal{V}_j^i \Big) \bigcup \Big(\bigcup_{j\in \mathrm{Int}^i} \mathcal{D}^i \setminus \mathcal{V}_j^i \Big) \right] \quad (4)$$

where Ext^i and Int^i are the sets of exterior contours and interior contours respectively in image i. The above expression is equivalent to (3) but it simplifies the function that updates depth intervals. Note here that when applying the above expression, the contribution of a viewing cone complement along the viewing line should be limited to the line interval visible from its image. Figure (4) displays the algorithm result for a synthetic object.

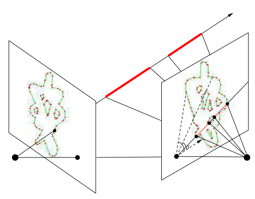


Figure 3: Contribution intervals (in red) to the visual hull surface along the viewing line. Epipolar line angles can be used to accelerate the search for the segments intersecting the epipolar line.

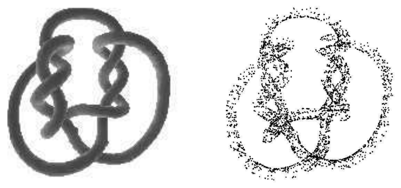


Figure 4: (a) the *knots* taken from Hoppe's web site [10]; (b) its visual hull surface points for 40 viewpoints located on a circle around the object.

3.3 Complexity

Assume that n, m and q are the number of images, contours per image and points per contour respectively, then the above algorithm computes $\theta(nmq)$ 3D points in $O(n^2m^2qr)$ time, where r is the upper bound complexity of the line-contour intersection function (step 6 in the algorithm) [1]. A naive implementation leads to $r = O(q)$ if we consider that occluding contours are polygons with q vertices on average. The overall asymptotic complexity would then be $O(n^2m^2q^2)$, or $O(N^2)$ where N is the number of 3D points computed.

[1]We suppose that the number of intersections between the epipolar line and an occluding contour is negligible compared to n, m and q, we thus expect the function that updates depth intervals to use $O(1)$ time.

Interestingly, the complexity r can be reduced to $O(1)$ by optimizing the intersection function. To this purpose, and between steps 2 and 3 of the algorithm, the image k can be rectified so that epipolar lines become horizontal lines (i.e., the epipolar rectification). In that case, search for intersections between the epipolar line l and the occluding contour O_l^k is simplified by using image ordinates as lookup values. Only the contour segments for which the epipolar line ordinate falls within the vertices' ordinates are to be considered. Equivalently, angles of lines joining the epipole and the contour vertices can also be used as lookup values, with the advantage that image rectifications are not required (see figure 3). Both solutions lead to $r = O(1)$ but add $\theta(n^2mq)$ operations to either rectify image coordinates or compute angle values of contour vertices. Note that in [17, 16] line slopes are used for similar reasons, however slopes do not always partition the image plane in a consistent way. In particular, the slope function is not monotonic when two successive contour vertices lie on both sides of the vertical line incident to the epipole, and hence such a function can not be used for lookup. Using the optimized intersection function, the asymptotic complexity reduces to $O(n^2m^2q)$.

4. Visual hull surface

We have shown in the previous section how to compute points on the surface of the visual hull. The next step is concerned with the estimation of the visual hull shape. Classical volumetric approaches consist in carving a partition of the 3D space into regular cells: the voxels. In contrast to this, our space partition lies on the computed visual hull points, and is thus composed of non-regular cells: the Delaunay tetrahedrons. The major advantage is to allow precision at a reasonable cost in time and space complexities.

4.1 Point triangulation

The approach we propose is based on the Delaunay tetrahedrization of the visual hull surface points. Delaunay triangulations have been widely used to reconstruct surfaces from unorganized 3D points. Indeed, this problem has received a lot of attention over the last decade and most of the proposed methods consider that the surface solution is included into the Delaunay triangulation of the input points. There are two advantages to the Delaunay triangulation: first, it ensures a regular partition of space in which cells satisfy properties such as having empty circumscribed balls; second fast and robust implementations exist.

The problem we address is similar except that the input data includes, in addition to the 3D points, the 2D image information. Thus, our approach also searches for a subset of the Delaunay triangulation, but the criterion applied to carve, or sculpt, the tetrahedral cells takes this additional information into consideration. In terms of complexity, the Delaunay tetrahedrization is known to have a worst case running time in $O(n^2)$ where n is the number of points. In our case, and as explained in the previous section, the number of 3D points is $\theta(nmq)$ where n, m and q are the number of images, contours per image and points per contour respectively. Thus the worst case complexity would be $O(n^2m^2q^2)$, which is more than the time required to compute the visual hull surface points. However, recent works (see [1] for instance) tend to show that the complexity of the Delaunay triangulation for points on a polyhedron is linear in the number of points. This is also confirmed by our experiments which show that most of the running time is spent in the previous step of the algorithm when computing visual surface points. Observe that the overall complexity is therefore not dominated by the Delaunay triangulation.

4.2 Surface extraction

The Delaunay triangulation leads to a set of tetrahedrons which form the convex hull of the set of input points. From this set of tetrahedrons, those contributing to the complement of the visual hull need to be identified and eliminated. A straightforward approach consists in computing the projections of their centroids onto the images and to check whether they lie inside any silhouette. Such an approach is fast if binary images representing background and foreground information are available, which is often the case with silhouette-based applications. It also gives satisfactory results as shown in the next section. Notice that more complex operations involving surface or volume criteria could also be applied. We are currently conducting experiments in that direction. The set of tetrahedrons whose centroids project inside the silhouettes are therefore considered as the visual hull cells. This set is not necessarily bordered by a manifold surface since the elimination may leave isolated tetrahedrons. Such tetrahedrons are detected in a final step where the triangular facets delimiting the visual hull tetrahedrons are identified. The final surface is thus a manifold composed of triangular facets such that all the vertices project onto occluding contours.

Remark also that most of the 3D points computed are naturally grouped pairwise since they delimit intervals. Thus, in addition to the 3D points, the segments defined by these pairs of points also form elements of the visual hull surface and should, therefore, be included into the space partition. To take these new elements into consideration, we have experimented a conforming Delaunay triangulation algorithm [7] which ensures that the triangulation includes any predefined linear complex (edges and faces). However, this algorithm adds a possibly important number of points to the input set, in order to satisfy the edges or faces constraints. Moreover it appears to be very slow and cancels the interest of a fast triangulation, which is a strong limitation especially in the case of real time applications. We also investigate alternative solutions to enforce these constraints.

5. Experimental results

We have experimented the described method on several input sets. A first experiment on the knot object compares our approach with a voxel-like approach. Figure 5 shows results obtained with the same silhouettes (40 images). The boundaries of the voxel grid were chosen close to the object, which is rarely the case in practical situations. Note that results are geometrically better with our approach for a fairly lower complexity. Indeed 3772 points are present in our model while 60^3 voxels must be verified in each image, not mentioning any surface extraction step, with the voxel approach. The number of images, however, has a linear influence on the upper bound complexity of the volumetric approach while it has a quadratic influence on the upper bound complexity of our approach. This is because the number of images does not affect the space partition with volumetric approaches while it does with our approach. However adding images does not always improve the estimated shape as shown by the next experiments.

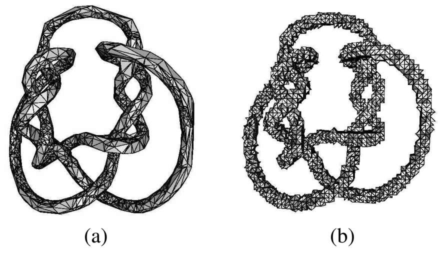


Figure 5: The visual hull surface of the knots: (a) our algorithm result (3772 points reconstructed) (b) a voxel like reconstruction with a $60^3 = 216000$ voxel grid.

The second set of experiments show results on a synthetic torus with cameras randomly distributed on a spherical region surrounding it. Figure 6 displays different visual hulls of the torus obtained with different numbers of points on each contours and different numbers of cameras. Observe that the running time of the algorithm is $O(n^2m^2q)$ where n is the number of images and q the number of points on each contour. Thus adding points on contours has less effect on the running time. Note also that the accuracy of the visual hull decreases surprisingly when the number of images reaches a certain value. This is especially clear in the left column, from 16 to 32 images. Such a behavior is explained by the fact that when adding images, visual hull points closer to the surface are also added. Thus, some of the Delaunay tetrahedrons get closer to the surface and their centroids may project outside some silhouettes. To avoid this behavior, the number of contour points should also be increased when increasing the number of images. Interestingly, this suggests that there is an optimal ratio between the number of images and the number of points on the contours.

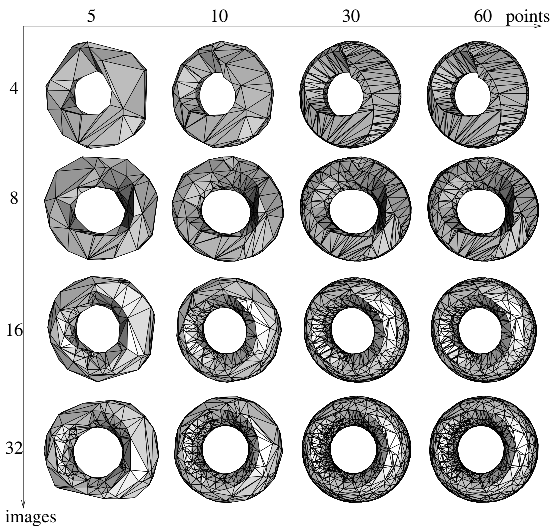


Figure 6: Visual hulls of a torus: the number of points per contour versus the number of images. Points are regularly sampled on the contours and images are randomly chosen on a sphere surrounding the torus.

Figure 7 shows results obtained with real objects. Contours are extracted in the images using the optimal algorithm described in [8]. The human example is interesting since it shows virtual objects as explained in section 2. We are also experimenting the same scenario with a cluster of PCs in real time situations, our preliminary results show that real time computations of fairly precise models can be expected.

6. Conclusion

In this paper, we have presented an approach for computing the visual hull of complex scenes when silhouettes are available. Our main contribution is to propose a hybrid algorithm which takes advantage of both volumetric approaches and surface-based approaches. The algorithm first computes points on the surface of the visual hull, and second extracts the visual hull surface from a Delaunay triangulation. The first step is achieved by computing points delimiting the visual hull complement. The second step is achieved by

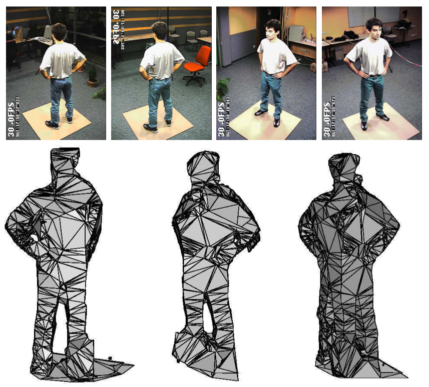


Figure 7: The visual hull of a person from 4 viewpoints. It contains all the geometric information available from the silhouettes. Nevertheless, note, in the right view, the virtual legs that are added to the person. This is due to the camera positions in this particular case and illustrates the point we made in section 2

taking the surface delimiting the polyhedrons that project inside the silhouettes. We have shown that our approach is equivalent to volumetric approaches for efficiency. They are both based on the same definition for visual hulls and they both use all the geometric information available from the silhouettes. However, we have also shown that our approach gives significantly better results in terms of precision, together with lower time and space complexities. The resulting reconstruction method is naturally aimed at real time modeling applications.

We are currently working on further improvements and applications of our method. First, tetrahedrons are classified according to the positions of their centroid projections in the images. This can be improved by applying other elimination schemes. Second, the Delaunay triangulation is applied on points only, however there are more information about the visual hull yet to be used, namely the contribution intervals along viewing lines. The final model could therefore be improved by including these intervals. Third, real time implementations of modeling methods are still a challenging issue, in particular when considering a cluster of PCs.

References

[1] Dominique Attali and Jean-Daniel Boissonnat. A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. Rapport de recherche 4453, INRIA, 2002.

[2] B.G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, 1975.

[3] E. Boyer and M.-O. Berger. 3D surface reconstruction using occluding contours. *IJCV*, 22(3):219–233, 1997.

[4] K.M. Cheung, T. Kanade, J.Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *CVPR'00*, volume 2, pages 714 – 720.

[5] C.H. Chien and J.K. Aggarwal. Volume/surface octress for the representation of three-dimensional objects. *CVGIP*, 36(1):100–113, 1986.

[6] R. Cipolla and A. Blake. Surface Shape from the Deformation of Apparent Contours. *IJCV*, 9:83–112, 1992.

[7] David Cohen-Steiner, Eric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3d. In *Proc. 18th Annu. ACM SoCG*, 2002.

[8] I. Debled-Rennesson and J.P. Reveillès. A linear algorithm for segmentation of digital curves. *IJPRAI*, 9(4):635–662, 1995.

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

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

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

[12] K. Kutulakos and S. Seitz. A Theory of Shape by Space Carving. *IJCV*, 38(3):199–218, 2000.

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

[14] S. Lazebnik, E. Boyer, and J. Ponce. On How to Compute Exact Visual Hulls of Object Bounded by Smooth Surfaces. In *CVPR'01*, volume I, pages 156–161.

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

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

[17] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image Based Visual Hulls. In *Siggraph*, pages 369–374, 2000.

[18] W. Niem. Automatic Modelling 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.

[19] 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.

[20] S. Sullivan and J. Ponce. Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs using Triangular Splines. *ICCV'98*.

[21] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP*, 58(1):23–32, 1993.

6.3 Visual shapes of Silhouette Sets

Marc Lapierre, Jean-Sébastien Franco and Edmond Boyer

Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization

Chapel Hill (USA), June 2006

Visual Shapes of Silhouette Sets

Jean-Sébastien Franco Marc Lapierre Edmond Boyer
GRAVIR–INRIA Rhône-Alpes
655, Avenue de l'Europe, 38334 Saint Ismier, France
{franco,lapierre,eboyer}@inrialpes.fr

Abstract

Shape from silhouette methods are extensively used to model dynamic and non-rigid objects using binary foreground-background images. Since the problem of reconstructing shapes from silhouettes is ambiguous, a number of solutions exist and several approaches only consider the one with a maximal volume, called the visual hull. However, the visual hull is not always a good approximation of shapes, in particular when observing smooth surfaces with few cameras. In this paper, we consider instead a class of solutions to the silhouette reconstruction problem that we call visual shapes. Such a class includes the visual hull, but also better approximations of the observed shapes which can take into account local assumptions such as smoothness, among others. Our contributions with respect to existing works is first to identify silhouette consistent shapes different from the visual hull, and second to give a practical way to estimate such shapes in real time. Experiments on various sets of data including human body silhouettes are shown to illustrate the principle and the interests of visual shapes.

1. Introduction

Recovering shapes from their projected contours in a set of digital images has been a subject of interest for the last three decades in the vision and graphics communities. The main interest of these contours is that they lead to region based modeling approaches which are rapid and do not rely on only local, and sensitive, photometric consistencies between images. They are therefore used to produce models, and especially initial models, in a number of modeling systems in particular dynamic systems which consider moving objects over time. Several methods have been proposed to solve the associated reconstruction problem among which one of the most successful is the *visual hull* [1, 14]. Such an approach consists in computing the maximal volume that projects inside image contours or, in other words, onto *silhouettes*. Straightforward approaches exist to this purpose [21, 18, 11], some of which are real time [6, 10]. While robust and easy to estimate, the visual hull is not, in general, a good geometric approximation of the observed shape. It can even be rather poor if a reduced number of views are considered. This is due to the fact that the visual hull is merely an extended bounding box, obtained by identifying the region in space where the observed shape can not be with respect to a set of silhouettes. Such a conservative approach does not report on shapes that are consistent with a given set of silhouettes, but on the union of the regions occupied by all such shapes. As a consequence, a number of viewpoints are required to refine this region and ensure that it is reasonably close to the observed object shape.

However, even a few silhouettes provide strong geometric information on shapes under little assumptions. Our intention in this paper is therefore to find better approximations of an object shape given its silhouettes while keeping the ability to model in real time. To this purpose, we introduce the *Visual Shapes* of a set of silhouettes, which are silhouette consistent shapes in the sense that their projected silhouette boundaries, with respect to given viewpoints, match the given silhouette contours. Beside the definition which helps in characterizing silhouette based models, often incorrectly considered as visual hulls in the literature, the main interest of visual shapes is to yield estimations more precise than visual hulls.

While the literature on visual hulls and their computation is vast, less efforts have been devoted to silhouette consistent shapes inside the visual hull. In [7, 22], first solutions were proposed to determine, along viewing lines, single points of contacts with the surface, under local second order assumptions. The associated approaches assume some knowledge on extremal contour connectivities, as well as simple shape topologies, but they allow smooth surfaces to be reconstructed. Our work is founded on the same observation that viewing lines along silhouette contours, and thus the visual hull surface, are tangent to the observed object surface. Following also this observation, approaches [9, 13, 5, 17] exploit the duality that exists between points and planes in 3D space, and estimate the dual of the surface tangent planes as defined by silhouette contour points. However, these approaches do not account for the fact that surface points lie on known viewing lines, in known intervals, and suffer therefore from various singularities. Also the visual shapes represent a more general concept since a family of plausible shapes, including the visual hull, is defined.

In [15], the topological structure of the visual hull is made explicit in the case of smooth objects. In this work, the mesh describing the extremal contour connectivity on the object surface is called the *rim mesh* and its connection with the visual hull mesh is identified. Unfortunately, this theoretical contribution does not yield a practical method to estimate the rim mesh in general situations, in particular with shapes having complex topologies. Recently, [12] and [20] proposed approaches to estimate the rim mesh on the visual hull surface by adding a photometric consistency constraint. However the rim mesh is not always well defined due to self-occlusions and strong assumptions need to be made on the topology of the observed objects, as stated in [16].

Our strategy is different from the afore-mentioned works. We first define a family of shapes which are consistent with a given set of silhouettes, namely the visual shapes. For one set of silhouettes, the associated shapes differ then by their contact with viewing lines of silhouette contour points: from isolated points, as for extremal contours on the observed shape, to the maximal intervals of the visual hull. Visual shapes are reduced to a single element when an infinite number of viewpoints, outside the shape's convex hull, is considered. In that case visual shapes and the visual hull are equivalent to the original shape, minus its concavities. However, in the general case, additional information is required to identify a single visual shape. Several criteria can be used to that purpose. In this paper, we experiment a very general assumption of local shape smoothness which is true in most real situations. The interest is to provide an approximation of the observed shape which is better than the visual hull, while keeping its robustness advantage over most modeling approaches. Such an approximation is useful not only as a final model but also as the initial input data to several modeling applications including motion capture or model refinement.

The paper is organized as follows. In Section 2, geometric entities related to visual shapes are introduced. In Section 3, visual shapes are defined and illustrated. In Section 4, it is explained how to compute visual shapes, and results with real data are presented, before concluding in 5.

2. Preliminaries

Suppose that a scene, containing an arbitrary number objects, is observed by a set of calibrated pinhole cameras.

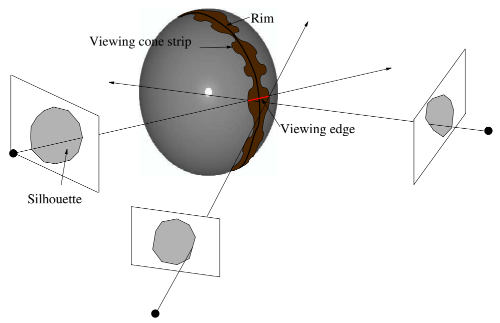


Figure 1. Viewing cone strip of a silhouette.

Suppose also that projections of objects in the images are segmented and identified as foreground. The foreground region of an image i consists then of the union of object projections in that image and, hence, may be composed of several unconnected components with non-zero genus. Each connected component is called a *silhouette*.

Consider the set of viewing rays associated with image points belonging to a single silhouette in one image. The closure of this set defines a cone in space, called *viewing cone*. The viewing cone delimiting surface is tangent to the surface of the corresponding foreground object along a curve called the *rim* (see figure 1). In what follows, we assume that a rim is formally defined as the locus of points on the object surface where viewing lines from one viewpoint are tangent to the surface.

The *visual hull* [1] is then obtained by intersecting viewing cones, possibly with respect to various image visibility domains [11]. It is a generalized polyhedron whose faces are made of cone patches, organized into strips with respect to silhouette contours.

A *viewing cone strip* corresponds then to contributions of a silhouette contour to the boundary surface of the visual hull (see figure 1). By construction, the rim associated with a silhouette contour lies inside the viewing cone strips associated to the silhouette. Observe that for non-smooth objects, the rim can become a strip itself within the viewing cone strip.

Of particular interest for this paper are *viewing edges*, corresponding to contributions of viewing rays to the visual hull surface. For one image point, such a contribution consists of one or several edges along the ray. A viewing cone strip can then be defined as the union of the viewing edges of the points on a silhouette contour. The viewing edges of an image point are easily obtained by finding silhouette contribution intervals along the point's viewing line, and computing then the common intersections of these intervals.

3. Definition

The visual hull is defined as the intersection of the viewing cones. As mentioned in the introduction, our objective is to identify a larger family of shapes associated to a given set of image silhouettes. To this purpose, we will focus on the part of the surface which is observed from silhouette contours, namely rims, and consider that shapes consistent with a set of silhouettes have rims with similar topologies. Hence, the proposed following definition:

Definition. *Let $\mathcal{S}$ be a set of scene silhouettes associated to a set of viewpoints $\mathcal{C}$. Then visual shapes $\mathcal{V}(\mathcal{S},\mathcal{C})$ of $\mathcal{S}$ and $\mathcal{C}$ are space regions V such that:*

1. *All rim points on the surface of V, belong to viewing cone strips of $\mathcal{S}$.*
2. *All viewing cone strips of $\mathcal{S}$ are tangent to V.*

The two above constraints ensure that, first, visual shapes are consistent with the given silhouettes, and second, that inside any viewing strip there is a rim. Such a definition yields a family of shapes which are consistent with silhouettes and viewpoints, i.e. all the volumes in space for which rims project onto given silhouettes and cover all of them. Intuitively, the visual shapes $\mathcal{V}(\mathcal{S},\mathcal{C})$ differ by the width, along viewing lines, of their rims, and identifying a single visual shape inside the solution family consists in deciding for the rim width based on *a priori* knowledge. Note that visual shapes include the visual hull as an extremal shape in the family that encloses all the others. We have then the following property:

Property . *Let $\mathcal{S}$ be a set of scene silhouettes associated to a set of viewpoints $\mathcal{C}$, then any viewing edge associated to contours points of $\mathcal{S}$ contains at least one point of any visual shapes $\mathcal{V}(\mathcal{S},\mathcal{C})$.*

This property means that all visual shapes are tangent to the visual hull surface along viewing cone strips. In particular, we expect better approximations of smooth shapes to be shapes with a single contact point with the visual hull surface along viewing lines. Visual shapes include shapes which satisfy that constraint. This will be used when computing visual shapes as explained in the next Section.

Visual shapes could also be seen as dual shapes of the visual hull, by the fact that they are shapes inside the visual hull with tangent contacts. However, the above definition is not restricted to a single shape but identifies a family of silhouette consistent shapes. Also in contrast to duality based approaches [5, 17], visual shapes are well defined shapes which do not suffer from singularities in generic situations. This is due to the fact that visual shape rim points are, by definition, on the visual hull surface which is itself well defined. In duality based approaches, estimated shapes do not necessarily satisfy this containment property since shape point locations are not restricted to viewing edges, or even viewing lines, but to planes. In that sense, visual shapes use all the information provided by silhouettes. The only assumption which is made so far is that observed shapes are tangent to all visual hull faces. Even if this is not always true, as shown in figure 2, it limits the reconstruction solution space in a reasonable way when no additional information are available to decide where the matter is.

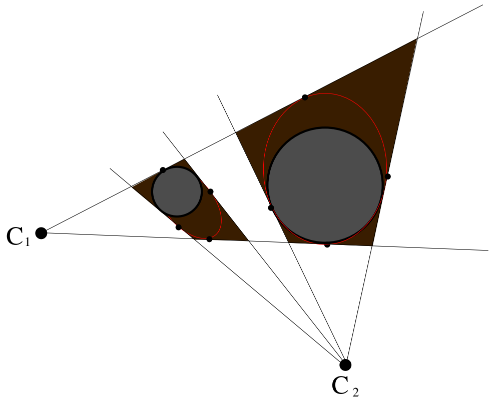


Figure 2. Cross Section of a situation where 2 cameras observe 2 spheres. In brown the resulting visual hull. In red, the surface of one of the associated visual shapes with single contact points locally with the visual hull surface. Observe that, by definition, the visual shape is tangent to the visual hull surface, but that the observed objects do not necessarily satisfy that property.

By definition, all visual shapes associated with a set of silhouettes share the same topology, that of the visual hull. Note however that the observed objects are not necessarily visual shapes of their silhouettes because of self-occlusions which can hide rim points and unoccupied visible space (see figure 2).

Figure 3 shows examples of visual shapes corresponding to silhouettes of a sphere. Sets with different numbers of silhouettes were used. The figure shows the visual hulls obtained with these sets as well as various visual shapes obtained by: (b) thinning viewing cone strips, (c) choosing a single contact point along viewing edges, and (d) estimating single contact point with local assumptions. Observe that in column (d), well delimited contours always appear on

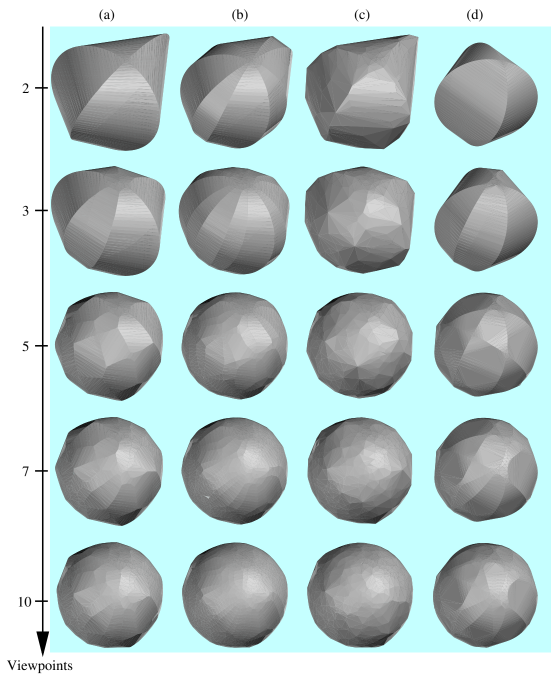


Figure 3. Visual shape examples with silhouette sets of a sphere. The top row correspond to silhouettes from 2 viewpoints, and the rows below show models obtained by progressively adding viewpoints. Columns are: (a) visual hulls; (b) visual shapes obtained by slightly thinning viewing cone strips; (c) visual shapes with one contact point, with the visual hull surface, randomly chosen inside viewing edges; (d) visual shapes with one contact point assuming the surface to be locally of order 2.

visual shapes. They correspond to the observed rims on the sphere. In that case, local assumptions about the observed surface are true, and all the estimated points inside viewing edges belong to the observed sphere. Note also that when increasing the number of views, visual shapes all converge to a single shape. With infinite viewpoints outside the scene convex hull, this *limit* shape becomes the observed shape from which concavities have been removed[1].

[1]This is the original definition of the visual hull by Laurentini[14]

4. Computation

In the previous Section, we introduced visual shapes of a set of silhouettes. These are shapes with the same topological rims with respect to the considered viewpoints. As a consequence, visual shapes of a set of silhouettes all have contributions inside viewing edges of silhouette contour points. Thus, the computation of visual shapes consists first in identifying these contributions inside the viewing edges, and second to estimate the surface connecting these contributions. This is described in the following Sections where we assume polygonal silhouette contours, as generally available in real situations.

4.1. Contributions along viewing edges

As mentioned earlier, viewing edges, or visual hull contribution intervals along viewing rays, are easily computed by intersecting ray projections with image silhouettes (see [4] for how to compute them efficiently). In figure 3, column (b) shows visual shapes obtained by thinning these viewing edges. This is a first solution, however this does not improve the estimation in a significant way with respect to the visual hull. As shown in column (c)-(d) of figure 3, a better estimation is related to the fact that viewing rays along silhouette contours only graze the surface at isolated points. This is true for smooth surfaces, but not only: even if the surface is locally planar, viewing rays will still be tangent at isolated points, except in the specific case where the viewing point belongs to the surface plane.

In the following, we thus assume a single contact point inside viewing edges. To identify the location of the contact point, different assumptions can be made. In [12] and [20], image photo-consistency assumptions are made to determine rim points inside visual hull faces. However photo-consistency applies to true surface points, and in numerous situations where self-occlusions occur there is no such point inside viewing edges, as explained before and shown for instance in figure 2. A shape estimated this way would still be a visual shape by definition, but with an unpredictable local behavior. Another possibility is to assume that the surface is locally of order 2, thus with a predictable local behavior. It is more or less the assumption made in duality based approaches [13, 5, 17] where the surface is assumed to be locally a quadric, or where finite differences are used to estimate derivatives. Our approach differs by the fact that we constrain the points we estimate to be inside well defined intervals along viewing rays, namely viewing edges. In contrast, duality based approaches estimate points dual to planes, and, importantly, can not guarantee that these points belong to the visual hull.

Another advantage of viewing edges is that they naturally define a local neighborhood through epipolar corre-

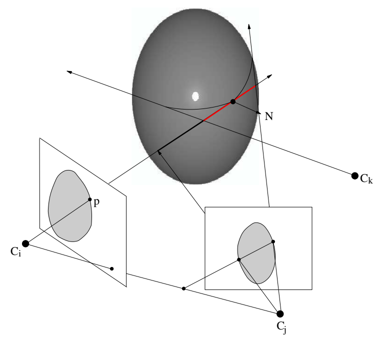


Figure 4. Viewing edges of point p in image i are delimited by viewing rays of epipolar correspondents of p. The neighborhood defined around P by these two correspondents is used to estimate local surface properties in the viewing direction. Note that when p moves along the silhouette contour, the viewing points C_j and C_k change when p reaches rim intersections on the surface.

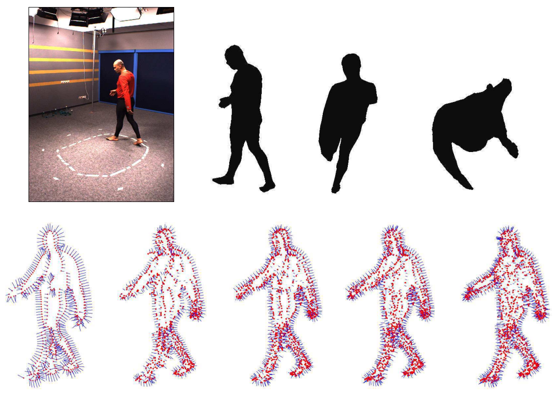


Figure 5. Visual shape points and normals under the assumption that the surface is locally of order 2. On top: one of the image used and 3 of the 6 silhouettes available. Bottom: estimated points (red) and normals (blue) with 2 (left) up to 6 (right) viewpoints.

spondences (see figure 4). Their boundary, i.e. the interval boundary points along viewing rays, identifies the epipolar correspondents, over all input silhouettes, such that the interval, where a surface point can lie, is minimal and not infinite in general[2]. Local neighborhoods defined in this manner are optimal for local estimation of surface properties. Using instead the epipolar parametrization between silhouettes, as in [7, 22, 3] and more recently in [17], does not ensure such a property since correspondences between silhouettes are imposed: points on silhouette at time t are matched with points on silhouettes at time $t \pm \epsilon$, and other silhouettes are not considered. Intervals along viewing rays defined by such correspondences can be infinite even when the visual hull is finite, hence making local surface estimations very difficult.

Each viewing edge defines a neighborhood composed of two epipolar correspondents. Thus for each viewing edge, we have three viewing rays which are locally tangent to the visual shape: 2 viewing lines from the epipolar correspondents and the viewing line supporting the viewing edge. From these three tangents, it is easy to estimate the position of the visual shape point inside the viewing edge, under the assumption that the surface is locally of order 2. To this purpose, we use the algorithm presented in [3]. This algorithm exploits the fact that the three viewing rays define locally two curves on the visual shape surface which present the same normal curvature at the contact point, and a linear solution for the surface point position inside a viewing edge exists.

[2]Viewing edge intervals are finite as long as the visual hull is finite.

Examples

Figure 5 illustrates the above estimation with silhouettes obtained in real conditions. Visual shape points, and their normals to the surface, are shown. Surface normals were classically computed as the cross products of viewing directions and tangents to the silhouette contours in the images. Note in this figure that even with two viewpoints, useful visual information can still be computed from silhouettes. The information computed this way, even if partial, can be useful for various applications. We have in particular successfully used such information, i.e. point locations and surface normals, as input data to a model based motion capture system [19].

4.2. From viewing edge contributions to shapes

In the previous section, we explained how to estimate viewing ray contributions to visual shapes. Several approaches were mentioned, from viewing edge thinning, to single contact point estimations. All these approaches allow visual shape points to be estimated, as well as their normals to the visual shape surface. However, a crucial issue is how to find the visual shape surface interpolating these points.

In the case of the visual hull, the associated mesh is completely defined from silhouette contours. It corresponds to a polyhedral mesh with a constant valence equal to 3[11] and its computation can be achieved from image primitives. For other visual shapes, no a priori information is available apart from their organization into strips onto the surface. However, this information only yields surface patches which are difficult to connect so as to form a valid shape. In [17], a solution is proposed which consists first in re-sampling rims according to parallel slicing planes, and second to solve the simpler problem of surface reconstruction from polygonal contours, for which standard tools exist. While robust, this solution can not guarantee precision since re-sampling introduces errors, nor can it guarantee that the estimated shape has a topology consistent with the observations, since the surface estimation is achieved without any consideration to the image information.

To compute a shape which interpolates visual shape points while being consistent with silhouettes, we use a fairly efficient solution based on the Delaunay tetrahedrization. This has been explored in the case of visual hulls[4] and we extend the idea to general visual shapes. The proposed method computes the Delaunay tetrahedrization of the visual shape points, then carves tetrahedrons of the resulting set, which project outside any image silhouette. Visual shapes are then the union of the tetrahedrons consistent with all the input silhouettes. While simple, the approach still raises a few issues to be discussed:

1. Often tetrahedrons do not project entirely inside or outside a silhouette. To decide whether a tetrahedron is inside or outside a silhouette, we sample several points inside the tetrahedron and verify their projection status with respect to the silhouette. The ratio of points inside and outside the silhouette is then considered for the decision. Another possibility would also be to subdivide the tetrahedron into sub-tetrahedrons and to carve the subdivision.

2. Carving must be achieved with some care if a manifold surface is expected. In some local configurations, tetrahedrons should not be carved to preserve local surface connectivity. These configurations have been identified in [2].

3. The Delaunay tetrahedrization does not necessarily reflect known connections inside viewing cone strips. This is not a critical issue in most cases but yields sometimes annoying visual artifacts in the computed model. To overcome this, a first solution consists in adding vertices to the silhouette polygonal contours, increasing therefore the probability that contour connections appear in the triangulation. While satisfying in most situations, this solution does not give any guaranty. One could therefore prefer using a conformal Delaunay tetrahedrization [8], which can ensure that the computed complex includes any predefined rim edges, with however a much higher computational complexity.

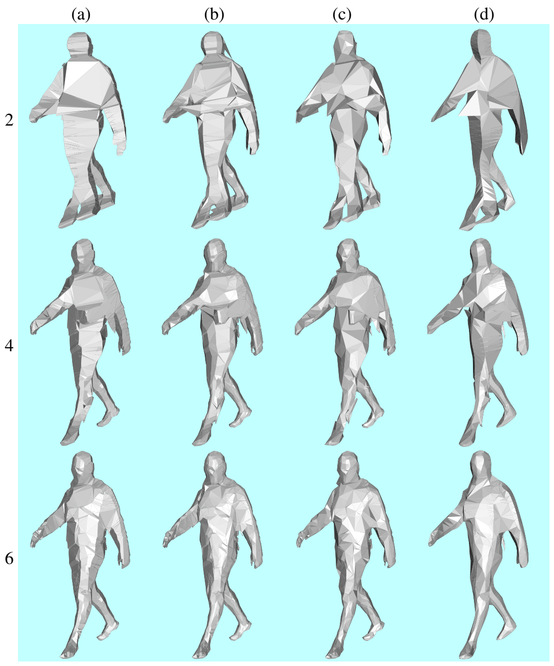


Figure 6. Visual shapes of a body shape with, from top to bottom 2, 4 and 6 viewpoints, and from left to right: (a) visual hulls; (b) thinned viewing cone strips; (c) random single contact point inside viewing edges; (d)single contact point with local smooth assumptions as described in Section 4.1.

Examples

Figure 6 illustrates the method with the same input data than in figure 5. Visual shapes were computed in a way similar to figure 3. In the top row, the visual shapes present a different topology than the human body, because too few viewpoints are used. It shows that visual shapes cover all the visible space, which is a reasonable behavior when no additional information about shape location is available. Note also that since the observed body model has a mostly smooth surface, the visual shapes with a local second order surface model, in column (d), are the most realistic estimations.

4.3. Texturing Visual Shapes

One of the strong advantage of visual shape models is that they project exactly onto silhouettes in the image, allowing therefore the photometric information inside silhouettes to be entirely mapped onto the model. However some difficulties remain in particular how to map textures on the model ? To this purpose, we developed an original approach. The idea is to consider each camera as a light source and to render the model using a shading model. The contribution of a view to the model textures is then encoded in the illumination values of the light source with the same location than the original view (see figure 7). These contribution values are then combined with texture values to obtain a final image. Depending on the shading model, purely diffuse or with specular like effects, texture mapping will be view-dependent or not. Though simple, this approach appears to be very efficient to texture models in real time (see the video submitted).

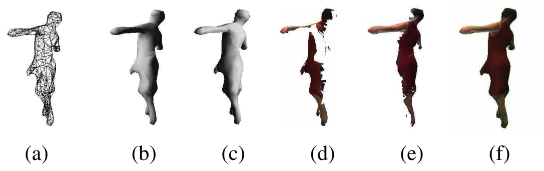


Figure 7. Texturing visual shapes: (a) the mesh, (b)-(c) the mesh rendered from 2 camera viewpoints, (d)-(e) the corresponding view contributions to the textured model, (f) the combined contribution of the the 2 camera images.

Figure 8 shows textured models of visual shapes similar to those used in the previous examples. When textured, all visual shapes share, by construction, the same appearance from viewpoints close to those of the acquisition process. Differences appear, and increase, when moving away from these viewpoints. Then, the visual hull reveals its bounding box like aspect, as viewing cone intersection curves become visible. This effect is made more obvious when considering dynamic visual shapes over time sequences. In that case, differences between chosen visual shapes are more visible, and our visual system naturally considers shapes with more likely local properties, e.g. the right model in figure 8, as more realistic.

5. Conclusion

We have introduce the visual shapes, which are a class of silhouette consistent shapes. The concept is useful to characterize shapes that project onto a set of silhouettes, and which are not necessarily the well known visual hull. This is especially useful when observing shapes with known properties, e.g. smoothness, since local assumptions can easily be used to identify and construct the most appropriate visual shape among a set of solutions. We have proposed an approach to compute points of the visual shape's surface, which are then used to compute this surface. The approach is robust and has been validated over various data sets, showing the interest of the method, in particular when modeling smooth surfaces such as human bodies. Issues we are currently considering include consistency of visual shapes over time sequences, and how to adequately account for photometric information.

References

[1] B.G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, pages 589–596, 1975.

[2] J.-D. Boissonnat. Geometric structures for three-dimensional shape reconstruction. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[3] E. Boyer and M.-O. Berger. 3D Surface Reconstruction Using Occluding Contours. *International Journal of Computer Vision*, 22(3):219–233, 1997.

[4] 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.

[5] 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, (USA)*, 2004.

[6] 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 2, pages 714 – 720, 2000.

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

[8] David Cohen-Steiner, Eric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3d. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 2002.

[9] G. Cross and A. Zisserman. Quadric Reconstruction from Dual-Space Geometry. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, (India)*, 1998.

[10] J.-S. Franco, C. Menier, E. Boyer, and B. Raffin. A Distributed Approach for Real-Time 3D Modeling. In *IEEE CVPR Workshop on Real-Time 3D Sensors and their Applications, Washington DC*, July 2004.

[11] J.S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In *Proceedings of the 14th British Machine Vision Conference, Norwich, (UK)*, 2003.

[12] Y. Furukawa and J. Ponce. Carved visual hulls for high-accuracy image-based modeling. In *technical sketch at SIGGRAPH 2005*.

[13] K. Kang, J.P. Tarel, R. Fishman, and D. Cooper. A Linear Dual-Space Approach to 3D Surface Reconstruction from Occluding Contours. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, (Canada)*, 2001.

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

[15] 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, 2001.

[16] Svetlana Lazebnik. Projective Visual Hulls. Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[17] 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)*, 2005.

[18] 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.

[19] M. Niskanen, E. Boyer, and R. Horaud. Articulated motion capture from 3-d points and normals. In Torr Clocksin, Fitzgibbon, editor, *British Machine Vision Conference*, volume 1, pages 439–448, Oxford, UK, September 2005. BMVA, British Machine Vision Association.

[20] S. Sinha and M. Pollefeys. Multi-view Reconstruction using Photo-consistency and Exact Silhouette Constraints: A Maximum-Flow Formulation. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, (China)*, 2005.

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

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

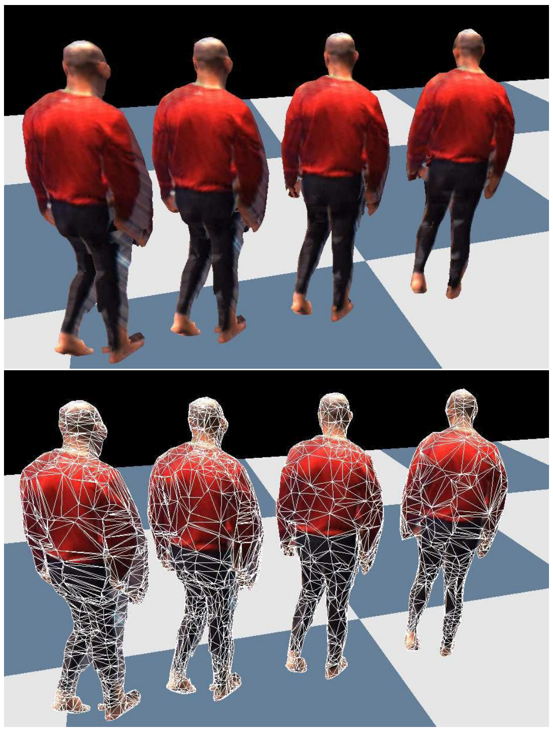


Figure 8. Visual shapes rendered with textures. From left to right: the visual hull, a model with thinned viewing cone strips, a model with random single contact points inside viewing edges and a model with single contact points satisfying local smooth assumptions.

6.4 On Using Silhouettes for Camera Calibration

Edmond Boyer

Proceedings of the 7th Asian Conference on Computer Vision, Hyderabad (India)

Hyderabad (India), January 2006

On Using Silhouettes for Camera Calibration

Edmond Boyer

MOVI - Gravir - INRIA Rhône-Alpes, Montbonnot, France
`Edmond.Boyer@inrialpes.fr`

Abstract. This paper addresses the problem of camera calibration using object silhouettes in image sequences. It is known that silhouettes encode information on camera parameters by the fact that their associated viewing cones should present a common intersection in space. In this paper, we investigate how to evaluate calibration parameters given a set of silhouettes, and how to optimize such parameters with silhouette cues only. The objective is to provide on-line tools for silhouette based modeling applications in multiple camera environments. Our contributions with respect to existing works in this field is first to establish the exact constraint that camera parameters should satisfy with respect to silhouettes, and second to derive from this constraint new practical criteria to evaluate and to optimize camera parameters. Results on both synthetic and real data illustrate the interest of the proposed framework.

1 Introduction

Camera calibration is a necessary preliminary step for most computer vision applications involving geometric measures. This includes 3D modeling, localization and navigation, am-ong other applications. Traditional solutions in computer vision are based on particular features that are extracted and matched, or identified, in images. This article studies solutions based on silhouettes which do not require any particular patterns nor matching or identification procedures. They represent therefore a convenient solution to evaluate and improve on-line a camera calibration, without the help of any specific patterns. The practical interest arises more specifically in multiple camera environments which are becoming common due, in part, to recent evolutions of camera acquisition materials. These environments require flexible solutions to estimate, and to frequently update, camera parameters, especially because often calibrations do not remain valid over time.

In a seminal work on motion from silhouettes, Rieger [1] used *fixed points* on silhouette boundaries to estimate the axis of rotation from 2 orthographic images. These fixed points correspond to epipolar tangencies, where epipolar planes are tangent to the observed objects' surface. Later on, these points were identified as *frontier points* in [2] since they go across the frontier of the visible region on a surface when the viewpoint is continuously changing. In the associated work, the constraint they give on camera motion was used to optimize essential matrices. In [3], this constraint was established as an extension of the traditional epipolar constraint, and thus was called the *generalized epipolar constraint*. Frontier points give constraints on camera motions, however they must first be localized on silhouette boundaries. This operation appears to be difficult:

© Springer-Verlag Berlin Heidelberg 2006

in [4] inflexions of the silhouette boundary are used to detect frontier points from which motion is derived, in [5] infinite 4D spaces are explored using random samples and in [6] contour signatures are used to find potential frontier points. All these approaches require frontier points to be identified on the silhouette contours prior to camera parameter estimation. However such frontier points can not be localized exactly without knowing epipoles. As a consequence, only approximated solutions are usually obtained by discrete sampling over a space of potential locations for frontier points or epipoles. We take a different strategy and bypass the frontier point localization by considering the problem globally over sets of silhouettes. The interest is to transform a computationally expensive discrete search into an exact, and much faster, optimization over a continuous space.

It is worth to mention also a particular class of shape-from-silhouette applications which use turntables and a single camera to compute 3D models. Such model acquisition systems have received noticeable attention from the vision community [7, 8, 9]. They are geometrically equivalent to a camera rotating in a plane around the scene. The specific constraints which result from this situation can be used to estimate all motion parameters. However, the associated solutions do not extend to general camera configurations as assumed in this paper.

Our approach is based first on the study of the constraint that both silhouettes and camera parameters must satisfy. We then derive two criteria: a quantitative smooth criterion in the form of a distance, and a qualitative discrete criterion, both being defined at any point inside a silhouette. This provides practical tools to qualitatively evaluate calibrations, and to quantitatively optimize their parameters. It appears to be particularly useful in multiple camera environments where calibrations often change, and for which fast on-line solutions are required.

This paper is organized as follows. Section 2 recalls background material. Section 3 precises constraints and respective properties of silhouettes, viewing cones and frontier points. Section 4 introduces the distance between viewing cones that is used as a geometric criterion. Section 5 introduces the qualitative criterion. Section 6 shows results on various data before concluding in section 7.

2 Definitions

Silhouette: Suppose that a scene, containing an arbitrary number objects, is observed by a set of pinhole cameras. Suppose also that projections of objects in the images are segmented and identified as foreground. $\mathcal{O}$ denotes then the set of observed objects and $\mathcal{I}_{\mathcal{O}}$ the corresponding binary foreground-background images. The foreground region of an image i consists of the union of objects' projections in that image and, hence, may be composed of several unconnected components with non-zero genus. Each connected component is called a *silhouette* and their union in image i is denoted $\mathcal{S}_i$.

Viewing Cone: Consider the set of viewing rays associated with image points belonging to a single silhouette in $\mathcal{S}_i$. The closure of this set defines a generalized cone in space, called *viewing cone*. The viewing cone's delimiting surface is tangent to the surface of the corresponding foreground object. In the same way that $\mathcal{S}_i$ is possibly

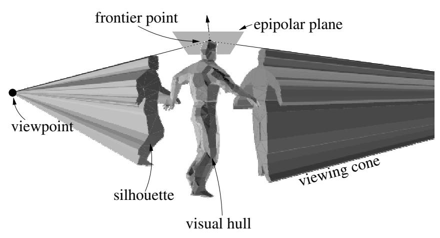


Fig. 1. A visual hull and 2 of its viewing cones

composed of unconnected components, the viewing cones of image i are possibly several distinct cones, one associated with each silhouette in $\mathcal{S}_i$. Their union is denoted $\mathcal{C}_i$. Note that individual objects are not distinguished here.

Visual Hull: The *visual hull* [10] is formally defined as the maximum surface consistent with all silhouettes in all images. Intuitively, it is the intersection of the viewing cones of all images (see figure 1). In practice, silhouettes are delimited by 2D polygonal curves, thus viewing cones are polyhedral cones and since a finite set of images are considered, visual hulls are polyhedrons. Assume that all objects are seen from all image viewpoints then:

$$\mathcal{VH}(\mathcal{I}_\mathcal{O}) = \bigcap_{i \in \mathcal{I}_\mathcal{O}} \mathcal{C}_i, \tag{1}$$

is the visual hull associated with the set $\mathcal{I}_\mathcal{O}$of foreground images and their viewing cones $\mathcal{C}_{i \in \mathcal{I}_\mathcal{O}}$. If all objects $\mathcal{O}$ do not project onto all images, then the reasoning that follows still applies to subset of objects and subsets of cameras which satisfy the common visibility constraint.

3 Geometric Consistency Constraint

In this section, the exact and optimal geometric consistency which applies with silhouettes is first established and its equivalence with more practical constraints is discussed.

3.1 Visual Hull Constraint

Calibration constraints are usually derived from geometric constraints reflecting geometric coherence. For instance, different image projections of the same feature should give rise to the same spatial location with true camera parameters. In the case of silhouettes, and under the assumption that no other image primitives are available, the only geometric coherence that applies comes from the fact that all viewing cones should correspond to the same objects with true camera parameters. Thus:

$$\mathcal{O} \subset \mathcal{VH}(\mathcal{I}_\mathcal{O}),$$

and consequently by projecting in any image i:

$$S_i \ \subset \ P_i(\mathcal{VH}(\mathcal{I}_\mathcal{O})), \ \forall i \in \mathcal{I}_\mathcal{O},$$

where $P_i()$ is the oriented projection[1] in image i. Thus, viewing cones should all intersect, and viewing rays belonging to viewing cones should all contribute to this intersection. The above expression is equivalent to:

$$\bigcup_{i \in \mathcal{I}_{\mathcal{O}}} [\mathcal{S}_i \; - \; P_i(\mathcal{VH}(\mathcal{I}_{\mathcal{O}}))] = \emptyset, \tag{2}$$

which says that the visual hull projection onto any image i should entirely cover the corresponding silhouette $\mathcal{S}_i$ in that image. This is the constraint that viewing cones should satisfy with true camera parameters. It encodes all the geometric consistency constraints that apply with silhouettes and, as such, is optimal. However this expression in its current form does not yield a practical cost function for camera parameters since all configurations leading to an empty visual hull are equally considered, thus making convergence over cost functions very uncertain in many situations. To overcome this difficulty, viewing cones can be considered pairwise as explained in the following section.

3.2 Pairwise Cone Tangency

We can easily derive from the general expression (2) the pairwise tangency constraint. Substituting the visual hull definition (1) in (2):

$$(2) \; \Leftrightarrow \; \bigcup_{i \in \mathcal{I}_{\mathcal{O}}} [\mathcal{S}_i \; - \; P_i(\bigcap_{j \in \mathcal{I}_{\mathcal{O}}} \mathcal{C}_j)] = \emptyset.$$

Since projection is a linear operation preserving incidence relations:

$$(2) \; \Rightarrow \; \bigcup_{i \in \mathcal{I}_{\mathcal{O}}} [\mathcal{S}_i \; - \; \bigcap_{j \in \mathcal{I}_{\mathcal{O}}} P_i(\mathcal{C}_j)] = \emptyset.$$

Note that, in the above expression, the exact equivalence with (2) is lost since projecting viewing cone individually introduces depth ambiguities and, hence, does not ensure a common intersection of all cones as in (2). By distributive laws:

$$(2) \; \Rightarrow \; \bigcup_{(i,j) \in \mathcal{I}_{\mathcal{O}} \times \mathcal{I}_{\mathcal{O}}} [\mathcal{S}_i \; - \; P_i(\mathcal{C}_j)] = \emptyset. \tag{3}$$

Expression (3) states that all viewing cones of a single scene should be pairwise tangent. By pairwise tangent, it is meant that all viewing rays from one cone intersect the other cone, and reciprocally. This can be seen as the extension of the epipolar constraint to silhouettes (see figure 2). Note that this constraint is always satisfied by concentric viewing cones, for which no frontier points exist. Note also that if (3) and (2) are not strictly equivalent, they are equivalent in most general situations.

[1] i.e. a projection such that there is a one-to-one mapping between rays from the projection center and image points.

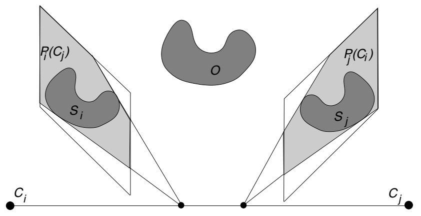


Fig. 2. Pairwise tangency constraint: silhouette $\mathcal{S}_i$ is a subset of the viewing cone projection $P_i(\mathcal{C}_j)$ in image i

3.3 Connection with Frontier Points

A number of approaches consider frontier points and the constraints they yield on camera configurations. Frontier points are particular points which are both on the objects' surface and the visual hull, which project onto silhouettes in 2 or more images, and where the epipolar plane is tangent to the surface (see figure 1). They satisfy therefore what is called the generalized epipolar constraint [3]. They allow hereby projective reconstruction when localized in images [5, 6]. The connection between the generalized epipolar constraint and the pairwise tangency constraint (3) is that the latter implies the former at particular frontier points. Intuitively, if two viewing cones are tangent then the generalized epipolar constraint is satisfied at extremal frontier points where viewing lines graze both viewing cones.

4 Quantitative Criterion

The pairwise tangency is a condition that viewing cones must satisfy to ensure that the same objects are inside all cones. In this section, we introduce a distance function that evaluates this condition.

4.1 Distances Between a Viewing Ray and a Viewing Cone

The distance function between a ray and a cone that we seek should preferably respect several conditions:

1. It should be expressed in a fixed metric with respect to the data, thus in the images since a 3D metric will change with camera parameters.
2. It should be a monotonic function of the respective locations of ray and cone.
3. It should be zero if the ray intersect the viewing cone. This intersection, while apparently easy to verify in the images, requires some care when epipolar geometry is used. Figure 3 depicts for instance a few situations where the epipolar line of a ray intersects the silhouette, though the ray does not intersect the viewing cone. These situations occur because no distinction is made between front and back of rays.
4. It should be finite in general so that situations in figure 3 can be differentiated.

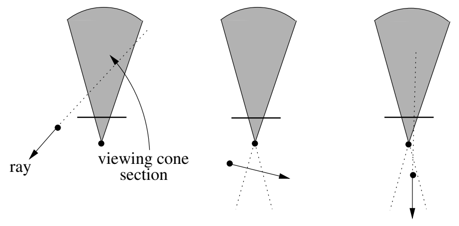


Fig. 3. A ray and the cross-section of the viewing cone in the corresponding epipolar plane. 3 of the situations where unoriented epipolar geometry will fail and detect intersections.

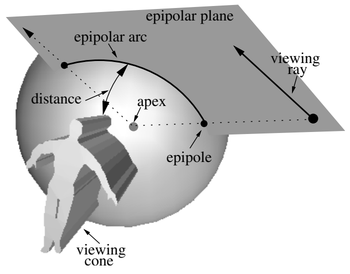


Fig. 4. The spherical image model: viewing rays project onto epipolars arcs on the sphere

In light of this, a fairly simple but efficient approach is to consider a spherical image model instead of a planar model (see figure 4), associated to an angular metric. The distance from a ray to a viewing cone is then the shortest path on the sphere from the viewing cone to the ray projection. This projection forms an epipolar circle-arc on the sphere delimited by the epipole and the intersection of the ray direction with the sphere. The ray projection is then always the shortest arc between these 2 points, which can coincide if the ray goes trough the viewing cone apex. Two different situations occur depending on the respective positions of the ray epipolar plane and the viewing cone:

1. The plane intersects the viewing cone apex only, as in figure 4. The point on the circle containing the epipolar arc and closest to the viewing cone must be determined. If such point is on the epipolar arc then the distance we seek is its distance to the viewing cone. Otherwise, it is the minimum of the distances between the arc boundary points and the viewing cone.
2. The plane goes through the viewing cone. The distance is zero in the case where the ray intersects the viewing cone section in the epipolar plane, and the shortest distance between the epipolar arc boundary points and the viewing cone section in the other case. This distance is easily computed using angles in the epipolar plane.

4.2 Distance Between 2 Viewing Cones

A distance function between a ray and a viewing cone has been defined in the previous section, this section discusses how to integrate it over a cone. The distance between

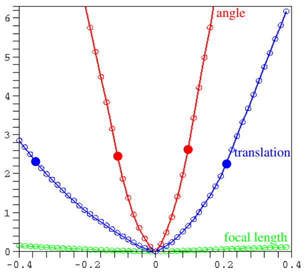


Fig. 5. The distance between 2 viewing cones as a function of: (green) one focal length which varies in the range $[f-0.4f, f+0.4f]$, with f the true value; (blue) one translation parameter to which is added from -0.4 to 0.4 of the camera-scene distance; (red) one Euler orientation angle which varies in the range $[\alpha-0.4\pi, \alpha+0.4\pi]$ with α the true value. The filled points denote the limit distances on curves above which the 2 cones do not intersect at all.

2 viewing cones is then simply defined by a double integration over the 2 concerned cones.

Recall that silhouettes and viewing cones are discrete in practice and thus defined by sets of contour points in the images and boundary rays in space. The simplest solution consists then in summing individual distances over boundary rays. Assume that r_i^k is the k^{th} ray on the boundary of viewing cone C_i, and $d(r_i^k, C_j) = d_{ij}^k$ is the distance between r_i^k and $\mathcal{C}_j$ as defined in the previous section. Then the distance D_{ij} between $\mathcal{C}_i$ and $\mathcal{C}_j$ is:

$$D_{ij} = \sum_k d_{ij}^k + \sum_l d_{ji}^l = d_{ij} + d_{ji}. \quad (4)$$

Remark that $D_{ij} = D_{ji}$ but $d_{ij} \neq d_{ji}$. The above expression is easy to compute once the distance function is established. It can be applied to all boundary viewing rays, however mainly rays on the convex hulls of silhouettes are concerned by the pairwise tangency constraint, we thus consider only them to improve computational efficiency. Figure 5 illustrates the distance D_{ij} between 2 viewing cones of a synthetic body model as a function of various parameters of one cone's camera. This graph demonstrates the smooth behavior of the distance around the true parameter values, even when the cones do not intersect at all.

5 Silhouette Calibration Ratio

Following the quantitative criterion, we introduce a simple qualitative criterion which evaluates how silhouettes contribute to the visual hull for a given calibration.

Recall that any viewing ray, from any viewing cone, should be intersected by all other image viewing cones, along an interval common to all cones. Let ω_r be an interval along ray r intersected by viewing cones, and let us call $\mathcal{N}(\omega_r)$ the number of image contributing (image for which a viewing cone intersects ω_r) inside that interval. Then the sum over the rays r: $\sum_r \max_{\omega_r}(\mathcal{N}(\omega_r))$, should theoretically be equal to $m(n-1)$ if m rays and n images are considered. Now this criterion can be refined by considering each image contribution individually along a viewing ray. Let ω_r^i be an interval, along ray r, where image i contributes. Then the silhouette calibration ration C_r defined as:

$$C_r = \frac{1}{m(n-1)^2} \sum_r \sum_i \max_{\omega_r^i}(\mathcal{N}(\omega^i)), \tag{5}$$

should theoretically be equal to 1 since each image should have at least one contribution interval with $(n-1)$ image contributions. This qualitative criterion is very useful in practice because it reflects the combined quality of a set of silhouettes and of a set of camera parameters. Notice however that it can hardly be used for optimizations because of its discrete, and thus non-smooth, nature.

6 Experimental Results

The pairwise tangency presented in the previous section constraint camera parameters when a set of static silhouettes $\mathcal{I}_\mathcal{O}$ is known. For calibration, different sets $\mathcal{I}_\mathcal{O}$ should be considered. They can easily be obtained, from moving objects for instance, as in [5]. The distances between viewing cones are then minimized over the camera parameter space through a least square approach:

$$\hat{\theta}_{\mathcal{I}_\mathcal{O}} = \min_\theta \sum_{(i,j)\,\in\,\mathcal{I}_\mathcal{O}\times\mathcal{I}_\mathcal{O}} D_{ij}^2, \tag{6}$$

where θ is the set of camera parameters to be optimized. $\hat{\theta}_{\mathcal{I}_\mathcal{O}}$ is equivalent to a maximum likelihood estimate of the camera parameters under the assumption that viewing rays are statistically independent. The above quantitative sum can be minimized by standard non-linear methods such as Levenberg-Marquardt.

6.1 Synthetic Data

Synthetic sequences, composed of images with dimensions 300×300, were used to test the approach robustness. 7 cameras, with standard focal lengths, are viewing a running human body. All camera extrinsic parameters and one focal length per camera, assuming known or unit aspect ratios, are optimized. Different initial solutions are tested by adding various percentages of uniform noise to the exact camera parameters. For the focal lengths and the translation parameters, the noise amplitudes vary from 0% up to 40% of the exact parameter value; for the pose angle parameters, the noise amplitudes vary from 0% up to 40% of 2π. Figure 6 shows, on the left, the silhouette calibration ratios after optimization; and on the right, relative errors in the estimated camera parameters after optimization using 5 frames per cameras. These results first validate the silhouette calibration ratio as a global estimator for the quality of any calibration with respect to silhouette data. Second, they show that using only one frame per camera is intractable in most situations. However, they prove also that using several frames, calibration can be recovered with a good precision even far from the exact solution. Other

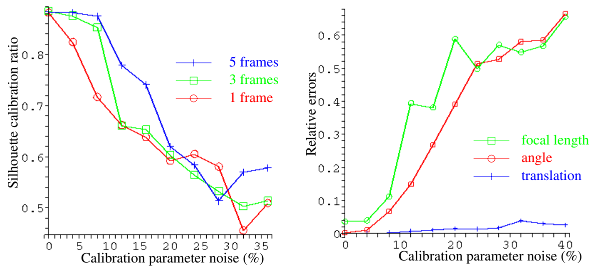


Fig. 6. Robustness to the initial calibration: right, the silhouette calibration ratio; left, the relative errors in the estimated camera parameters for the 5 frame case: errors relative to the true value for the focal length, errors relative to the distance camera-scene for the translation parameter and errors relative to π for the angle parameter

experiments, not presented due to lack of space, show that adding a reasonable amount of noise to silhouette vertices, typically a 1 pixel Gaussian Noise, only slightly changes these results.

6.2 Real Data

Our approach was also tested in a real environment with 6 firewire cameras viewing a moving person. A calibration obtained by optimizing an initial solution using known points is available and will be considered as the ground truth. In the following experi-

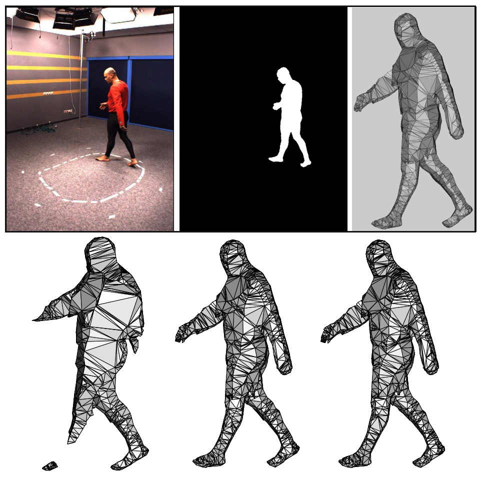


Fig. 7. Top, one of the original image, the corresponding silhouette and the visual hull model obtained with ground truth calibration. Bottom, 3 models which correspond to calibrations obtained with our method and using respectively 1, 3 and 5 frames per camera.

ments, we use the same initial solution for the calibration with viewing cones. As for the synthetic case, all camera extrinsic parameters and one focal length per camera are optimized. Figure 7 shows, on top, the input images and a visual hull model obtained using ground truth values for calibration. In the bottom, models obtained from the same silhouettes, but using our approach with respectively 1, 3 and 5 frames per camera. Apart from a scale difference, not shown and due to the fact that fixed dimensions were imposed for the ground truth solution, the 2 most-right models are very close to the ground truth one.

7 Conclusion

We have studied the problem of estimating camera parameters using silhouettes. It has been shown that, under little assumptions, all geometric constraints given by silhouettes are ensured by the pairwise tangency constraint. A second contribution of this paper is to provide a practical criterion based on the distance between 2 viewing cones. This criterion appears to be efficient in practice since it can handle a large variety of camera configurations, in particular when viewing cones are distant. It allows therefore multi-camera environments to be easily calibrated when an initial solution exists. The criterion can also be minimized using efficient and fast non-linear approach. The approach is therefore also aimed at real time estimation of camera motions with moving objects.

References

1. Rieger, J.: Three-Dimensional Motion from Fixed Points of a Deforming Profile Curve. Optics Letters **11** (1986) 123–125
2. Cipolla, R., strm, K., Giblin, P.: Motion from the Frontier of Curved Surfaces. In: Proceedings of 5th International Conference on Computer Vision, Boston (USA). (1995) 269–275
3. Åström, K., Cipolla, R., Giblin, P.: Generalised Epipolar Constraints. In: Proceedings of Fourth European Conference on Computer Vision, Cambridge, (England). (1996) 97–108 Lecture Notes in Computer Science, volume 1065.
4. Joshi, T., Ahuja, N., Ponce, J.: Structure and Motion Estimation from Dynamic Silhouettes under Perspective Projection. In: Proceedings of 5th International Conference on Computer Vision, Boston (USA). (1995) 290–295
5. Sinha, S., Pollefeys, M., McMillan, L.: Camera Network Calibration from Dynamic Silhouettes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington, (USA). (2004)
6. Furukawa, Y., Sethi, A., Ponce, J., Kriegman, D.J.: Structure from Motion for Smooth Textureless Objects. In: Proceedings of the 8th European Conference on Computer Vision, Prague, (Czech Republic). (2004)
7. Fitzgibbon, A., Cross, G., Zisserman, A.: Automatic 3d model construction for turn-table sequences. In: Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments. Volume 1506 of Lecture Notes in Computer Science. (1998) 154–170
8. Mendoça, P., Wong, K.Y., Cipolla, R.: Epipolar Geometry from Profiles under Circular Motion. IEEE Transactions on PAMI **23** (2001) 604–616
9. Jiang, G., Quan, L., Tsui, H.: Circular Motion Geometry Using Minimal Data. IEEE Transactions on PAMI **26** (2004) 721–731
10. Laurentini, A.: The Visual Hull Concept for Silhouette-Based Image Understanding. IEEE Transactions on PAMI **16** (1994) 150–162

6.5 Exact Polyhedral Visual Hulls

Jean-Sébastien Franco and Edmond Boyer

Proceedings of the 14th British Machine Vision Conference

Norwich (UK), September 2003

Exact Polyhedral Visual Hulls

Jean-Sébastien Franco and Edmond Boyer
INRIA Rhône-Alpes
ZIRST Montbonnot, 655, av. de l'Europe, 38334 St-Ismier CEDEX, France
jean-sebastien.franco@inrialpes.fr *edmond.boyer@inrialpes.fr*

Abstract

We propose an exact method for efficiently and robustly computing the visual hull of an object from image contours. Unlike most existing approaches, ours computes an exact description of the visual hull polyhedron associated to polygonal image contours. Furthermore, the proposed approach is fast and allows real-time recovery of both manifold and watertight visual hull polyhedra. The process involves three main steps. First, a coarse geometrical approximation of the visual hull is computed by retrieving its viewing edges, an unconnected subset of the wanted mesh. Then, local orientation and connectivity rules are used to walk along the relevant viewing cone intersection boundaries, so as to iteratively generate the missing surface points and connections. A final connection walkthrough allows us to identify the planar contours for each face of the polyhedron. Implementation details and results with synthetic and real data are presented.

1 Introduction

Visual hulls are object shape approximations which can be determined from object silhouettes in images. Such approximations capture all the geometric information given by the image silhouettes. Visual hulls are extensively used in a number of modeling applications including human modeling systems. Their popularity is largely due to the fact that straightforward approaches exist and are easy to implement. However, existing approaches are only partial solutions to the visual hull estimation problem and do not address all the essential criteria in modeling: exactness, robustness and fastness. Our motivation is therefore to propose an exact approach which computes the visual hull polyhedron associated to a finite number of discrete silhouettes in a fast and robust way.

Silhouettes were first considered by Baumgart [1] who proposed to compute polyhedral shape approximations by intersecting silhouette cones. The term visual hull was later coined by Laurentini [9] to describe the maximal volume compatible with a set of silhouettes. Following Baumgart's work, a number of modeling approaches based on silhouettes have been proposed. They can be roughly separated into two categories : volume based approaches and surface based approaches.

The first category includes methods that approximate visual hulls by collections of elementary cells called voxels. An early approach in this category was proposed by Martin and Aggarwal [11] who used parallelepipedic cells aligned with the coordinate axis. Later on, octrees were proposed [4] as adaptive data structures for representing visual hulls and efficient approaches [16, 13, 3] were presented to compute voxel-based representations.

See [14] and [7] for reviews on volume based modeling approaches. All these approaches are based on regular voxel grids and can handle objects with complex topologies. However, the space discretizations used lead to approximations only, with a poor precision to complexity trade-off.

As shown in [10], the visual hull surface is a projective topological polyhedron made of curve edges and faces connecting them. In the case of piecewise-linear image contours, it becomes a regular polyhedron. The second category of approaches estimates elements of this polyhedron by intersecting silhouette cones. This includes several works which focus on individual points reconstructed using local second order surface approximations, see [5] for a review. Approaches have also been proposed to compute surface patches[15], or individual strips [12] of the visual hull. In the latter work, the computed strips are exact parts of the visual hull, however the approach duplicates cone intersection operations and requires an additional step to connect these different parts, with no topological guarantee. We will show in this paper that the complete visual hull polyhedron can be recovered as a whole with a reduced number of operations. Most of the mentioned surface based approaches suffer from numerical instabilities around frontier points as identified in [10]. Consequently, they often lead to surface models which are incomplete or corrupted, in particular when considering objects with complex topologies.

The method that we propose follows recent work [2] which separates the visual hull computation into two steps. A first step computes viewing edges of the visual hull and a second approximates its faces through a Delaunay triangulation. Our approach improves this scheme to produce, with less time complexity, the exact polyhedron that is silhouette consistent. To this aim, we replace the second step mentioned above with an algorithm which straightforwardly recovers mesh connectivity. Our main contribution with respect to all the mentioned approaches is to provide an algorithm which is exact given polygonal silhouettes and low in time complexity.

The paper is organized as follows. Section 2 introduces definitions and describes how viewing edges are computed. Section 3 discusses the local polyhedron orientation properties relevant to our task. Section 4 describes how these properties are used to follow the cone intersections and generate the visual hull mesh. Section 5 presents our results and future work.

2 Preliminaries

2.1 Definitions

Assume that a scene, composed of several objects, is observed by a set of pinhole cameras. The objects' surfaces are supposed to be orientable closed surfaces, smooth or polyhedral with possibly non-zero genus. *Rims* are locus of points, on the object surface, where viewing rays are tangent to the surface. Rims project onto image curves, called the *occluding contours* [11], which border the object silhouettes in the image plane. Occluding contours are oriented in the images. Their orientation is such that the object silhouette lies on the left of the oriented contour. Hence, exterior contours are oriented counterclockwise and interior contours are oriented clockwise. We will call the *inside region* of an occluding contour the closed region of the image plane delimited by the contour and containing the silhouette, and we will call the *outside region* its complement in the image plane. Note that in the following, we will consider that occluding contours are polygonal contours.

A *viewing cone* is a generalized cone in $\mathbb{R}^3$ whose apex is the image center and whose base is the inside region of an occluding contour. More formally, the *viewing cone* V associated with the occluding contour O is the closure of the set of rays passing through points inside O and through the camera center. V is thus tangent to the corresponding object surface along the rim that projects onto O. According to the orientation of O, exterior or interior, the viewing cone V is an acute or obtuse cone of $\mathbb{R}^3$ respectively. Viewing cone boundaries intersect along space curves which do not lie on the surface, except at *frontier points* where rims intersect. Note that in the case of polyhedral surfaces, frontier points are not necessarily isolated points and can form frontier edges.

We assume that using some standard background subtraction pre-process, images are transformed into sets of silhouettes which are themselves sets of polygonal contours. Recall the topological nature of the *visual hull* [10], a projective structure, with *frontier points* where rims intersect; *triple points* where three cones intersect; *cone intersection curves*; and *strips* corresponding to possibly unconnected contour contributions to the visual hull surface. Visual hulls are usually defined as the intersection of the viewing cones associated to all image contours, however such an intersection must be performed in full consistency with the silhouette information. To this purpose, two definitions of the visual hull VH can actually be considered [2]:

$$VH = \bigcap_{Images} \Big(\bigcup_{Silhouettes} \Big(\bigcap_{Contours} V \Big)\Big), \tag{1}$$

or:

$$VH^c = \bigcup_{Images} \Big(\bigcap_{Silhouettes} \Big(\bigcup_{Contours} D \setminus V \Big)\Big), \tag{2}$$

where VH^c is the visual hull complement in $\mathbb{R}^3$, D is the image visibility domain in $\mathbb{R}^3$ and $D \setminus V$ is the complement of V relative to this domain. Considering the visual hull or its complement is equivalent since the surface of interest borders both regions. The visual hull complement is in fact what is implicitly considered when carving voxels with volumetric methods. The first definition above is equivalent to the set of points in $\mathbb{R}^3$ that project inside one silhouette in every image while the second limits the projection constraint to the images where the points are visible. They differ by the fact that objects under consideration should be seen in every image with the first definition but not necessarily with the second. Both definitions may add independent virtual objects that do not appear in the original scene, but another difference is that the second definition may add more virtual objects as a consequence of the projection constraint relaxation. Note that polygonal contours such as those we take as input induce a polyhedral visual hull. Our algorithm computes exactly this polyhedron, which we will later refer to as being the *exact visual hull* in the context of piecewise linear silhouettes.

2.2 Computing viewing edges

Viewing edges are intervals along viewing lines. They correspond to viewing lines contributions to the visual hull surface and are thus associated to image points on occluding contours. We use this as the input of the second step in our algorithm. There are several advantages in doing so: computing such a set of edges has proven to be fast, simple and well-defined, and has already been used in different reconstruction applications [12, 2, 3]. Also, since edges computed by this method are already part of the visual hull polyhedron, it is a greedy initialization step. We recall in this section how to compute them efficiently given polygonal occluding contours.

For each occluding contours vertices in each image, we can compute its viewing line. The viewing edges associated to this viewing line are then defined by the intervals of points which satisfy any of the definitions introduced in 2.1. These intervals can be determined iteratively, using intersections with the silhouettes in every other image. At each iteration, intervals are updated according to their intersections with contributions from new contours or images. Such operation can be easily achieved in 2D by means of the epipolar geometry as shown in fig. 1. Importantly, for each generated intersections, we choose to record what edges in the images generated each intersection.

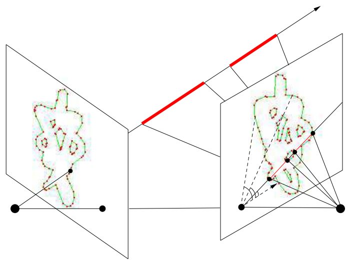


Figure 1: Viewing edges (in bold) along the viewing line. Epipolar line angles can be used to accelerate the search for the image segments intersecting the epipolar line.

Intervals along the viewing line are updated according to definition 1 or 2. A direct application of these definitions may not be straightforward since it requires contours to be grouped according to the silhouettes they belong to. A simpler solution takes advantage of the fact that interior contour contributions are unbounded along the viewing line. Thus, the union of an interior contour contribution with those of any disjoint exterior contour is equivalent to the identity for the former contribution. Following this principle, expressions 1 and 2 become:

$$VH = \bigcap_{Images} \left[(\bigcup_{Exteriors} V) \bigcap (\bigcap_{Interiors} V) \right], \tag{3}$$

and:

$$VH^c = \bigcup_{Images} \left[(\bigcap_{Exteriors} D \setminus V) \bigcup (\bigcup_{Interiors} D \setminus V) \right], \tag{4}$$

which require the contour orientations only. Note that in real situations, viewing cones are not necessarily tangent and hence, not all contour vertices give birth to viewing edges. The viewing edges computed during this initialization step form the initial seed for the subsequent steps in the algorithm.

2.3 Algorithm outline

The algorithm we propose consists in three main steps. The first step is to compute the viewing edges as explained above. However, this initial representation is incomplete. Typically, triple points, where three viewing cones intersect, project on each image contour at a point which is not necessarily one of the initial image contour vertices considered for viewing edge computation. As such, they are not part of the initial viewing edge vertices and need to be computed. Thus the goal of the second step is to recover all missing vertices and connections, so as to construct the complete oriented mesh describing the polyhedron. We have devised a scheme for following the cone intersection boundaries on the surface, using the geometry and orientation properties of the image contours discussed in section 3. With such properties, we can identify where connections are missing, and in what local direction to look for the next intersection boundary vertices. We can therefore iteratively generate the missing triple points, by detecting intersections with viewing cones, as described in section 4. As a third and final step, explained in section 4.3, the algorithm walks through the previously generated edge graph to identify each face.

3 Recovering the Local Orientation

3.1 Strip Orientation

The viewing edges we computed as a first step are a discrete representation of the visual hull strips (fig. 2a). We will not attempt to explicitly reconstruct frontier points or recover any topological features specific to theoretical visual hulls, as they do not extend to real visual hull surfaces as defined in section 2.1: because of inherently noisy calibrations and discretization steps, the perfect cone tangency that occurs at frontier points for theoretical visual hulls is lost. Note however that there still are regions of very close tangency, where both topology and orientation between strips may be complex and unstable.

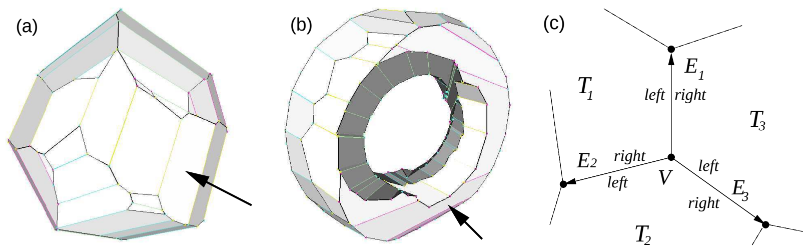


Figure 2: (a) Visual Hull of a sphere from 4 views as generated by our algorithm. Notice the strip structure and viewing edges, in light color. Symptoms of lost cone tangency: indicated strip covers all others, in a region where there should have been frontier points. (b) Visual hull of a torus from 4 images. Notice the branching of the indicated strip. (c) The relationship between vertex V's orientation (the ordering of pairs (E_i, T_i) around it) and orientation (left and right) of edges leaving from V on the visual hull polyhedron. Note that vertices are trivalent in the degenerate case, being the intersection of three planes.

Strips can have bifurcations (fig. 2b), and more generally can have several components, as soon as there are images with distinct silhouettes or hyperbolic contour portions. Nevertheless, each viewing edge we computed can be oriented with respect to the image it was backprojected from. We know that outer contours are oriented counter-clockwise and inner contours clockwise (see 2). For any contour vertex q in a given image, there are two edges e_1 and e_2 incident to q, respectively preceding and succeeding q according to the contour orientation. Note that any vertex q backprojects in space to a viewing line V_q. Similarly, any edge e of the contour backprojects into space to an infinite triangular planar patch T_e. The image contour orientation extends to these planar patches, and we can call *up* the direction on the strip induced by the positive direction of the corresponding image contour. Notice that with this definition of up and down on a strip, every viewing edge also inherits definitions for *front* and *back*, front always being the direction pointing toward the camera center. Moreover, since each viewing edge is defined by two points in space, these points can be labeled P_{front} and P_{back}. These image and strip orientability features are the foundations upon which we can build local surface orientation.

3.2 Polyhedron Point and Edge Orientation

In order to generate a consistent mesh, we must ensure that consistent orientation properties are propagated during processing. The initial stable and robust orientation property is the strip orientability described above. It is used as a basis to construct and propagate the local orientation for each primitive we will generate on the mesh. For oriented edges, this orientation information reduces to knowing what is left and what is right; for vertices, it reduces to knowing how incident edges are ordered around it.

Let us first consider an oriented edge of the visual hull polyhedron. Such an edge borders two visual hull surface regions, one locally on its left and the other locally on its right. These surface regions are a planar subset of the two corresponding triangular planar patches backprojecting from image edges, which we can label T_{left} and T_{right}. Furthermore, the edge itself represents the contribution segment to the visual hull of the line intersection between these two patches. Note that for the initial viewing edges, the (T_{left}, T_{right}) pair can be identified using the strip orientation properties (see 3.1): for example, if you consider the oriented edge $[P_{front}, P_{back}]$, what lies locally left of the edge is locally *up* on the strip. Reciprocally for oriented edge $[P_{back}, P_{front}]$, what lies locally left is locally *down* on the strip. Keeping track of this information for each oriented edge is also the key to maintain a consistent topological ordering between edges leaving from the same vertex, as illustrated in fig. 2c. If a vertex is visited coming from a given edge during a mesh walkthrough, it is then possible to query for existence of neighboring edges, and to make left or right turns at this vertex. We have therefore provided the necessary tools to generate and iterate over surface primitives.

4 Recovering the Missing Edges

We will now present the algorithm to follow cone intersection curves on the surface of the visual hull polyhedron, and generate the complete mesh of the visual hull as output. Existing surfacic approaches [12] have focused on the full edge plane intersections with the visual hull, which computes these curves as a side effect. However this requires numerous polygon intersection and transformation operations. Most edges and points of the intermediate polygons do not contribute to the final visual hull face. Following the intersection curves ensures that we will always focus on the relevant surface primitives, with an immediate complexity gain. Furthermore, all intersection operations in our algorithm resolve in the 2D image planes with one-parameter unknown computations.

Cone intersection curves materialize as a full graph of k triple points joining several viewing edge vertices, as illustrated in fig. 3a. We have developed a simple scheme to recover this graph, with two substeps. First, whenever a missing edge connection is identified on a polyhedron point, we must determine a direction in which the edge is leaving (see 4.1). Then, we use the direction's projection in images to detect triple points, and to identify which vertex this edge leads to (section 4.2).

4.1 Recovering the Direction of Missing Edges

Given any vertex V on the polyhedron, the vertex has three edge connections leaving from it, in the non-degenerate case. Let us focus (fig. 3b) on a particular type of vertex V found on the incomplete polyhedron surface, used as a *search seed*. Such a vertex already has one connected edge E, but two of its edges E_{left} and E_{right} are *potentially* still missing. Notice that E is considered oriented *toward* V; it is the direction of the walkthrough. E is fully known: we know what infinite backprojected planar patches T_{left} and T_{right} lie locally left and right of E. We also know what third planar patch T_{gen} generated the vertex V, through intersection with the two patches above. Notice that the initial viewing edges fit exactly this description, each of their two vertices being an initial search seed.

Now, we must recover the missing edge connections E_{left} and E_{right}. Since the search might have been launched from other connected branches of the graph, these edges might already exist. However, when missing, we must determine what vertex the connection leads to, in order to provide a complete edge description. Its nature (triple point or viewing edge vertex) and position must be determined. Solving both of these problems requires knowing in what 3D direction leaving from V this vertex lies, e.g. the *search half-line*. For example, if E_{left} is missing, we compute the half line as the intersection of planar patches T_{left} and T_{gen}, oriented left of E. Reciprocally we use T_{gen} and T_{right} if E_{right} is missing. We then apply the scheme described in the next section to each missing edge.

4.2 Identifying Missing Incident Vertices

We use an intersection search scheme to identify the vertex each missing edge leads to. The context of this search is illustrated in fig. 3c. Note that for both possible missing edges the local (T_{left}, T_{right}) pair is known: in fig. 3b, this pair corresponds to (T_{left}, T_{gen}) for missing edge E_{left}, and (T_{gen}, T_{right}) for E_{right}. For a given missing edge, consider the segment intersection between its two local T_{left} and T_{right} patches. The wanted edge is a subset of this segment, because this segment represents the contribution to the visual hull of two images only: the ones T_{left} and T_{right} backproject from. Use of more images reduces this contribution because of successive intersections. Nevertheless our two initial patches define natural search bounds, the brackets in fig. 3c. The lower bound is trivially given by the position of V, current search seed, and the upper bound by L, the most restrictive of the four viewing lines bordering patches T_{left} and T_{right}.

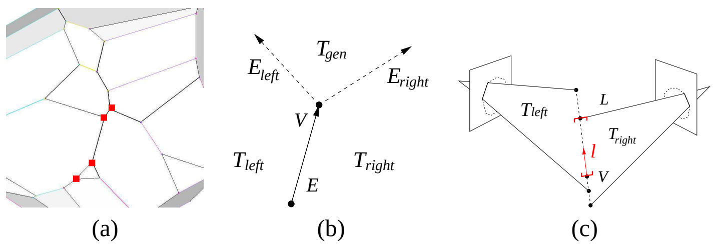


Figure 3: (a) Close-up view of a sphere's visual hull. Black edges are the discrete cone intersection curves, squares are triple points. (b) Spawning search directions from a given initial situation on the visual hull surface. (c) Once the search direction l is known for an identifi ed missing connection, the knowledge of the local T_{left} and T_{right} infi nite triangular patches results in natural search bounds for the incident vertex along this direction.

It is possible that no other image viewing cone intersects this segment, when its projection lies inside all other silhouettes. In this case these initial bounds exactly describe the wanted edge. Note that the incident vertex then lies on viewing line L, which defined the upper bound. Therefore, it has already been computed in the viewing edges stage, and its instance can be retrieved once identified among all viewing edge points of L. Possibly however, the viewing cone of a third image does intersect this segment, hence creating a triple point. Which is why every third possible image must be considered, in order to iteratively compute the most restrictive upper bound. We therefore reproject the search half-line in each possible third image, and compute the intersection with the silhouette, so as to update the bound. The nature of the incident vertex, viewing edge vertex or triple point, is known only after this process. Indexing triple points from the three planar patches which define it ensures that they are computed only once. Once identified, a newly generated triple point serves as a new search seed for the mesh generation; the process described in 4.1 is recursively applied to it. A search branch is completed once it meets an already computed vertex. We have therefore defined how to follow and generate the cone intersection segments starting with the initial viewing edge structure. The output is the complete polyhedron mesh, such that no polyhedron vertex is computed twice.

4.3 Identifying Faces of the Polyhedron

With the oriented edge computed in step two, we can now apply the third and final step: retrieving face information of the polyhedron. Note that each 2D edge in the initial images potentially contributes to the visual hull polyhedron surface, its contribution lying in its backprojected edge plane. In fact, this contribution is a single general planar polygon, with possibly several inside and outside contours, as noticed in [12]. With the vertex properties described in 3.2, graph walkthroughs can be constrained to follow a given backprojected edge plane and keep it locally left, by taking a left turn at each vertex. We also constrain the traversal of each polyhedron edge, in order to ensure that it will be walked through exactly twice, once in each direction. Hence an $O(v)$ walkthrough (with v the number of vertices in the final polyhedron) enables us to recover the entire contour set associated to every planar polygon previously mentioned. This is also an advantage of our method, the output polygons being described in their most general and concise form. Retrieving triangles for a specific application can be done efficiently by applying existing $O(v \log v)$ planar tessellation algorithms. Given the completeness of the mesh generated in step two, this constrained walkthrough ensures that our final polyhedron satisfies the manifold and watertight properties.

5 Results and Future Work

We have experimented with a preliminary implementation of our algorithm, on a variety of synthetic and real input sets for validation. Contours in each view are discretized to be used as input for our algorithm, using an optimal segment recognition algorithm [6]. This guarantees that the generated piecewise linear contours describe exactly the boundary pixels separating the background and foreground regions. Given that our algorithm computes the exact polyhedron associated to a given input contour set, the generated polyhedron is the best geometric information we can obtain after the background subtraction step.

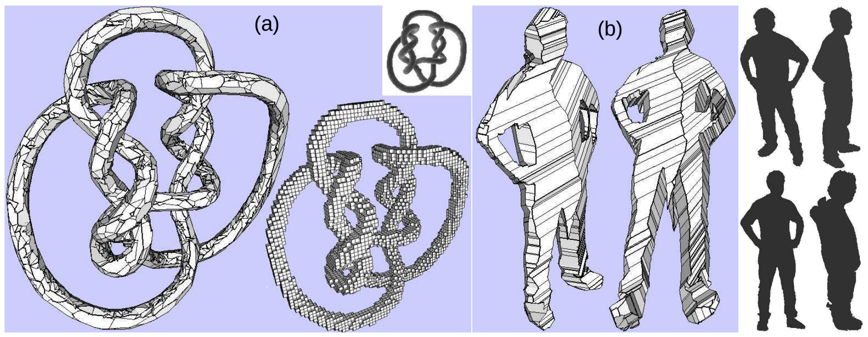


Figure 4: (a) Visual hull as obtained by our method, of the knots object taken from Hoppe's web site [8], from 42 viewpoints surrounding the object. Reconstruction of its 11146 points and 16719 edges took 12.8sec on a 1.8GHz PC. Discretization of silhouettes resulted in 200 contour points per image on average. Its voxel-based counterpart shows much less precision under the same conditions, using a well fitted $64^3 = 262144$ voxel grid. Original model is shown top right. (b) Two renderings of the visual hull of a human shape using 4 acquired 640x480 silhouettes (shown right). 2316 points, 2356 edges, computed in 142msec, with 250 contour points per image silhouettes.

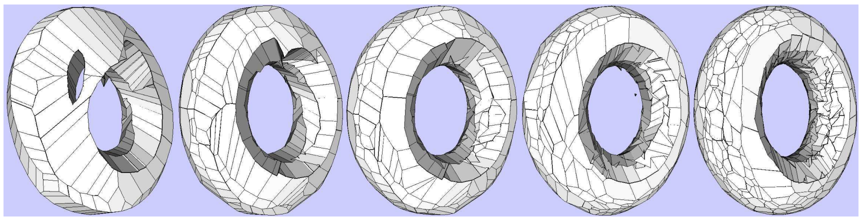
A first experiment places our algorithm in a complex topological situation (fig. 4a), while still yielding artifact-free results. A second experiment involves real image data from four cameras in a virtual reality studio. It results in a correct model, usable for such applications as scene relighting. We have also provided several visual hulls of a torus (fig. 5), which illustrate the impact of the number of views on model complexity. Computation times for this non-optimized version are of the order of 100ms for a dozen viewpoints, making it suitable for real-time applications. Implementation speedups are still possible. We will explore complexity reduction through simplification of the input contours. We will use this algorithm for real-time human modeling from video flows in virtual studios. Also, we will provide complexity estimations in the near future. Arguably this algorithm could be optimal for the visual hull polyhedron computation problem. Clearly it surpasses existing visual hull surface reconstruction algorithms: each operation in the algorithm is a greedy one, each computed primitive is known beforehand to be part of the visual hull, and uses the minimal necessary information to be determined, namely reprojection and intersection in images. Experimentally, the execution time and number of operations of this approach rivals with the voxel-based approaches, while attaining geometrical exactness hardly accessible to those methods.

References

[1] B.G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, 1975.

[2] E. Boyer and J.S. franco. A hybrid approach for computing visual hulls of complex objects. In *CVPR'03*, volume I, pages 695–701, 2003.

[3] 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 *CVPR'03*, volume II, pages 375–382, 2003.



4 / 714pts,19ms *8 / 1090pts, 75ms* *12 / 1410pts,125ms* *16 / 1656pts,217ms* *42 / 3126pts,1.44s*

Figure 5: Visual hulls of a torus under an increasing number of viewpoints, with 60 contour points per silhouette. Our method accounts for all available geometric information, with an increasingly complex and precise model. Very high numbers of viewpoints don't offer a good complexity/precision trade-off, since the visual hull inherently becomes more sensitive to image calibration and discretization noise. 10 to 20 viewpoints offer very decent results.

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

[5] R. Cipolla and P.J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, 1999.

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

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

[8] H.Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH'92*, volume 26(2), pages 71–78, 1992.

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

[10] 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.

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

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

[13] W. Niem. Automatic Modelling 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.

[14] 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.

[15] 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.

[16] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP*, 58(1):23–32, 1993.

6.6 Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid

Jean-Sébastien Franco and Edmond Boyer

Proceedings of the 10th International Conference on Computer Vision

Beijing (China), October 2005

Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid

Jean-Sébastien Franco Edmond Boyer
GRAVIR - INRIA Rhône-Alpes
655, av. de l'Europe, 38330 Montbonnot, France
firstname.lastname@inrialpes.fr

Abstract

In this paper, we investigate what can be inferred from several silhouette probability maps, in multi-camera environments. To this aim, we propose a new framework for multi-view silhouette cue fusion. This framework uses a space occupancy grid as a probabilistic 3D representation of scene contents. Such a representation is of great interest for various computer vision applications in perception, or localization for instance. Our main contribution is to introduce the occupancy grid concept, popular in the robotics community, for multi-camera environments. The idea is to consider each camera pixel as a statistical occupancy sensor. All pixel observations are then used jointly to infer where, and how likely, matter is present in the scene. As our results illustrate, this simple model has various advantages. Most sources of uncertainty are explicitly modeled, and no premature decisions about pixel labeling occur, thus preserving pixel knowledge. Consequently, optimal scene object localization, and robust volume reconstruction, can be achieved, with no constraint on camera placement and object visibility. In addition, this representation allows to improve silhouette extraction in images.

1. Introduction

Silhouette-based methods are popular for use in multi-camera environments mainly due to their simplicity and computational efficiency. These methods concern 3D modeling, multi-object localization and motion capture applications, among others. Often however in such methods, silhouettes of objects of interest are extracted using a binary labeling of pixels into foreground or background, for each view separately, and prior to any 3D operation. Unfortunately, such monocular labeling, called *background subtraction*, is difficult to achieve in a general and uncontrolled environment. Several reasons account for this, in particular perturbations due to: camera sensor noise, ambiguities between objects and background colors, changes in the lighting of the scene (including shadows of objects of interest), etc. In addition, monocular background labeling can dramatically alter 3D perception from multiple views in the presence of camera calibration errors, or if disparities between image acquisition times exist.

Our goal is therefore to find a representation of multi-view silhouette cues, where inference about silhouettes is of greater robustness to the aforementioned uncertainties than single view silhouette inference. Intuitively, the simultaneous knowledge of all images brings more information about silhouettes than knowledge from only one image. This idea has lead us to compute silhouette fusion in 3D space, in order to integrate the contribution of all images. The result of such fusion naturally encodes shape information. As such it can be used to improve many silhouette-based applications, from shape modeling to silhouette extraction, as we will show.

Very often silhouettes are used to infer shapes in a two-step process: an individual decision about silhouette occupancy is made on a per-view basis, then shape and position are inferred geometrically from all available silhouettes using *visual hull* methods [11]. These methods can lead to a surface representation of the objects of interest [5], a voxel representation [16], or image-based representation [13]. While visual hull estimation can be exact from a set of silhouettes [5], silhouette extraction methods come generally with several caveats resulting from the perturbations mentioned earlier. Our approach allows to delay the occupancy decision to a later stage and, as such, makes a better use of the available silhouette information.

Several methods have also been proposed to bypass silhouette estimation altogether, as many algorithms reconstruct the scene structure based only on photometric information [10]. Others possibly state it as the solution of a global state optimization problem: using level sets [4], or graph cuts [7]. Probability grid representations have already been used by the community, mainly to solve photometric problems[1]. These methods generally have high complexities and computational costs compared to silhouette methods, as they must deal with the visibility of points on the object's surface. This is why there are still many situations where silhouette methods are preferred (e.g. VR platforms, real-time setups), or used to initialize a more elaborate photometric method [9].

More closely related, Magnor *et al.* [7] solve a similar problem with two views using graph-cuts, where stereo disparity and silhouettes are simultaneously estimated. Zeng *et al.* propose a multi-view background silhouette extraction, based on a costly geometric scheme, with the additional constraint of common object visibility [18]. A similar idea for silhouette information integration has been proposed, using however a discrete formulation and a coarser image model [14]. Grauman *et al.* [8] propose a method to estimate the most probable multi-view silhouette set using a learned human silhouette prior, and therefore integrate a higher level of semantics, but with limited genericity. Robotics works from S. Thraun *et al.* [12] propose a solution for the closely related problem of object localization from a robot-acquired image sequence. These approaches solve silhouette-based problems in multi-view sequences with, however, limited application domains. Our approach is at a lower level, and is intended to enrich 2D silhouettes cues by embedding them into a 3D representation independently of the application.

We propose a new framework based on the occupancy grid: a voxel grid of object occupancy probabilities in space, associated to a sensor model. The occupancy grid has been extensively used in the robotics community [3], to represent a robot's environment for navigation, based on range sensor observations, with depth and orientation measurements. Our contribution is to extend the occupancy grid concept to image sensors, and to restate shape-from-silhouette estimation as a sensor fusion problem. To this extent, we provide each pixel with a *forward* sensor formulation which models the pixel observation responses to the voxel occupancies in the scene. Our formulation accounts for each pixel's visibility region, voxel sampling issues, small camera calibration errors, and sensor reliability. This model is in turn used to infer the answer to the more difficult inverse question: given the color observations, where is the matter located in the scene. We also show that the resulting occupancy grid can be used to perform multi-view background subtraction, where silhouette estimation in each view benefits from the knowledge of other views.

2. Problem Statement

We consider the problem of silhouette cue fusion from multiple views. We assume we are given a *current set* of images, obtained from fully calibrated cameras. We also assume that a set of *background images* of the scene, free from any *object of interest*, have previously been observed for each camera. Importantly, no assumption is made about the existence of a visibility domain common to all cameras.

The problem is formulated as the separate Bayesian estimation, for each voxel, of how likely it is occupied by an object of interest. We formulate the problem using a forward sensor model: we model the relationship from causes to observations. Namely, in our problem, we will model how a voxel influences image formation. This enables us, using Bayesian inference, to solve the more difficult inverse problem: express the voxel occupancy likelihood using images as a noisy measurement of scene state.

Solving a Bayesian problem requires computing the joint probability of all variables of interest (which we define in §2.1), prior to any inference. This joint probability distribution must then be decomposed and simplified, based on the main statistical dependencies we choose to consider between variables (§3 and §3.1). In particular, parametric forms must be assigned to the various terms of the decomposition to explicitly model the uncertain relationship between variables (§3.2 and §3.3). This simplifies the inference of voxel occupancy distributions, which are inferred from the joint probability expression using Bayes' rule (§4).

2.1. Main problem variables

We label the set of n current images as $\mathcal{I}$. $\mathcal{I}^i$, $i = 1 \cdots n$ is then the image data of camera i, and $\mathcal{I}^i_p$ is the image data at pixel p in image i, expressed in some color space (RGB, YUV, etc). Although not studied explicitly in this paper, additional image cues can be enclosed in the $\mathcal{I}^i_p$ term, such as the image gradient or some other local feature, without loss of generality. We assume that the image data of the corresponding m observed background images can be summarized into a single statistical model image $\mathcal{B}^i$, $i = 1 \cdots n$. Both image data sets are produced by n cameras with known projection matrices $\mathbf{P}^i$. τ symbolizes the prior knowledge we introduce into the model. This includes what we now about the scene, what we know about sensor characteristics, our general knowledge about the system.

We define $\mathcal{G}$ as our space occupancy grid. For each space point X in the grid discretization we associate the corresponding binary occupancy variable $\mathcal{G}_X \in \{0, 1\}$, respectively free or occupied. As a common occupancy grid assumption [3], we assume statistical independence between voxel occupancies, and compute each voxel occupancy likelihood independently for tractability. Results show that independent estimation, while not as exhaustive as a global search over all voxel configurations, still provides very robust and usable information, at a much lower cost.

We have defined our input and output variables. We now introduce an important hidden variable set per image, the silhouette detection maps $\mathcal{F}^i$, $i = 1 \cdots n$. These maps define, for each pixel p in image i, a binary silhouette detection variable $\mathcal{F}^i_p$. $\mathcal{F}^i_p = 1$ if the pixel sensor p in image i *reports* the presence of an object of interest anywhere along its viewing line. We insist on this definition, since there is a possibility that an object *is* indeed present along the viewing line of pixel p, but that the pixel sensor itself *fails* to detect and report this information for internal or external causes (modeling sensor failures will be discussed in §3.2). These detection maps represent the silhouette information in our model, over which we wish to marginalize.

3. Joint Probability Decomposition

Our goal is to infer the occupancy $\mathcal{G}_X$ of a voxel at position X, given $\mathcal{I}$, $\mathcal{B}$, and τ. Thus, we must first model the impact of $\mathcal{G}_X$ on the observations. Modeling the relationships between the variables involved requires computing the joint probability of these variables, $p(\mathcal{G}_X, \mathcal{I}, \mathcal{B}, \mathcal{F}, \tau)$. We propose the following decomposition, based on the statistical dependencies expressed in Fig. 1:

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

- $p(\tau)$, $p(\mathcal{B} \,|\, \tau)$ are the prior probabilities of our parameter set, and of background image parameters. Since we have no *a priori* reason to favor any parameter values, or background image configurations, we set these terms to a uniform distribution. They thus disappear from any subsequent inference.
- $p(\mathcal{G}_X \,|\, \tau)$ is the prior likelihood for occupancy, which could vary with X for example. We consider the occupancy to be at the top of the causality chain, thus the independence with all other variables except τ. We choose not to favor any voxel location and set this term to uniform, being mainly interested in the regularization of voxels induced by observations in this paper.
- $p(\mathcal{F} \mid \mathcal{G}_X, \tau)$ is the silhouette likelihood term. The dependencies considered reflect that voxel occupancy in the scene explains object detection in images.
- $p(\mathcal{I} \,|\, \mathcal{F}, \mathcal{B}, \tau)$ is the image likelihood term. Image colors are conditioned by object detections in images, and the knowledge of the background color model.

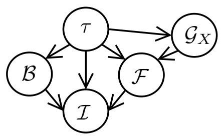


Figure 1. Variables of our system and their dependency graph. τ: prior knowledge we introduce in the model. $\mathcal{G}_X$: occupancy at voxel X. $\mathcal{B}$: background model maps. $\mathcal{F}$: silhouette detection maps. $\mathcal{I}$: observed images.

3.1. Sensor fusion simplifications

Pixel colors in input images are treated as noisy observations of the model. We consider that the noise is independently and identically distributed. Each pixel's color observation can be considered independent of all others, given the observation's main cause, the background data and silhouette detection state of the pixel: the image likelihood term can thus be simplified to a product of per pixel terms, $p(\mathcal{I} \,|\, \mathcal{F}, \mathcal{B}, \tau) = \prod_{i,p} p(\mathcal{I}_p^i \,|\, \mathcal{F}_p^i, \mathcal{B}_p^i, \tau)$.

All pixel detections can also be considered independent, given the knowledge of their main cause, namely the voxel occupancy. The silhouette likelihood is therefore similarly simplified: $p(\mathcal{F} \mid \mathcal{G}_X, \tau) = \prod_{i,p} p(\mathcal{F}_p^i \mid \mathcal{G}_X, \tau)$. Thus, the joint probability distribution of variables of interest reduces to the following product of per pixel terms:

$$p(\mathcal{G}_X, \mathcal{I}, \mathcal{B}, \mathcal{F}, \tau) = \prod_{i,p} p(\mathcal{F}_p^i \,|\, \mathcal{G}_X, \tau) p(\mathcal{I}_p^i \,|\, \mathcal{F}_p^i, \mathcal{B}_p^i, \tau) \quad (1)$$

We have therefore reduced the evaluation of the joint probability of all variables to two much friendlier subproblems. First, expressing the likelihood of silhouette detection at a single pixel, given the knowledge of our voxel's occupancy (§3.2). Second, expressing the likelihood of the color observation at a single pixel, given the silhouette detection state, and background color information at this pixel (§3.3). We now focus on these two terms.

3.2. Silhouette Formation Term

The silhouette detection likelihood $p(\mathcal{F}_p^i \mid \mathcal{G}_X, \tau)$ models the silhouette detection response of a single pixel sensor (i, p) to the occupancy state of our voxel of interest $\mathcal{G}_X$. We need to introduce two local hidden process variables $\mathcal{S}$ and $\mathcal{R}$ to balance the influence of this voxel. Fig. 2 introduces the variables and statistical dependencies of this subproblem. In an ideal and noiseless setup, the two variables $\mathcal{F}_p^i$ and $\mathcal{G}_X$ would be self-sufficient and the relationship between them expressed as simple logic: if our voxel X is occupied, and if it projects to pixel p, then silhouette detection occurs at pixel p, $\mathcal{F}_p^i = 1$. This is the implicit formulation used by all classical visual hull methods.

However, there are sources of uncertainty which perturb this intuitive reasoning. First, the assumption that a voxel lies on the viewing line of a pixel is itself uncertain. This can be due to many external causes: potential camera calibration errors, camera mis-synchronization, which both introduce misalignment in the scene. Voxel sampling is also an issue, since no voxel perfectly projects to a pixel, and its projected surface can cover several. Second, there can be causes for sensor detection other than the voxel itself: an object occupancy other than the one related by $\mathcal{G}_X$, or a change in background scene appearance (an *internal* sensor failure due to the nature of the sensor model).

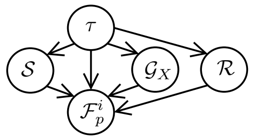

$$\begin{tikzcd} & \tau \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\ \mathcal{S} \arrow[dr] & & \mathcal{G}_X \arrow[dl] & \mathcal{R} \arrow[dll] \\ & \mathcal{F}_p^i & & \end{tikzcd}$$

Figure 2. Variables and dependency graph of the per-pixel silhouette detection subproblem. τ: prior knowledge. $\mathcal{G}_X$: voxel occupancy. $\mathcal{S}$: sampling variable. $\mathcal{R}$: external detection cause. $\mathcal{F}_p^i$: silhouette detection at pixel (i,p).

Modeling these hidden causes is possible using two boolean random variables, the *sampling variable* $\mathcal{S}$ and *external detection cause variable* $\mathcal{R}$. This leads to two expressions for the silhouette detection prior $p(\mathcal{F}_p^i \,|\, \mathcal{G}_X, \tau)$. First, let us consider the case where our voxel X is known to be occupied ($\mathcal{G}_X = 1$):

$$\begin{aligned} p(\mathcal{F}_p^i \,|\, [\mathcal{G}_X\!=\!1], \tau) &= p(\mathcal{S}\!=\!0 \,|\, \tau)\, \mathcal{U}(\mathcal{F}_p^i) \\ &+ p(\mathcal{S}\!=\!1 \,|\, \tau)\, \mathcal{P}_d(\mathcal{F}_p^i) \end{aligned} \tag{2}$$

By definition, $\mathcal{S}$=1 if voxel X is on the viewing line of pixel (i,p). When this is not the case ($\mathcal{S}$=0), the knowledge of our voxel's occupancy is irrelevant to sensor detections at this pixel, thus the uniform distribution $\mathcal{U}(\mathcal{F}_p^i)$ for silhouette detection in (2). If the voxel is on the viewing line of p ($\mathcal{S}$=1), then detection at the pixel is ruled by the probability distribution $\mathcal{P}_d(\mathcal{F}_p^i)$. In practice we set this distribution using a constant $P_D \in [0,1]$, which is a parameter of our system: $\mathcal{P}_d([\mathcal{F}_p^i = 1]) = P_D$ is the detection rate of a pixel sensor, and $\mathcal{P}_d([\mathcal{F}_p^i = 0]) = 1 - P_D$ is its detection failure rate. Detection failure occurs when the pixel sensor relates that there is no matter on the viewing line, when in fact there is. This is useful to our problem: sometimes silhouette extraction fails locally. Accounting for this uncertainty gives our model a chance to still recover the correct voxel information thanks to contributions of other images.

Let us now consider the case where our voxel is known to be empty ($\mathcal{G}_X = 0$):

$$\begin{aligned} p(\mathcal{F}_p^i \,|\, [\mathcal{G}_X\!=\!0], \tau) = p(\mathcal{S}\!=\!0 \,|\, \tau)\ \mathcal{U}(\mathcal{F}_p^i) \\ + p(\mathcal{S}\!=\!1 \,|\, \tau)\ \big[\ p(\mathcal{R}\!=\!1 \,|\, \tau)\ \mathcal{P}_d(\mathcal{F}_p^i) \\ + p(\mathcal{R}\!=\!0 \,|\, \tau)\ \mathcal{P}_f(\mathcal{F}_p^i)\ \big] \end{aligned} \tag{3}$$

Still, no knowledge can be inferred about detection when the voxel is not on the viewing line of p ($\mathcal{S}\!=\!0$). Yet in the case where voxel X is on p's viewing line ($\mathcal{S}\!=\!1$), we cannot yet draw conclusions about its detection state. By definition, $\mathcal{R}$=1 accounts for the possibility that some other object lies on the same viewing line as the voxel: in this case detection is again ruled by the distribution $\mathcal{P}_d(\mathcal{F}_p^i)$. However, in the case no other object obstructs the viewing line ($\mathcal{R}$=0), detection is ruled by distribution $\mathcal{P}_f(\mathcal{F}_p^i)$. We set this distribution using a constant $P_{FA} \in [0,1]$, a parameter of our system: $\mathcal{P}_f([\mathcal{F}_p^i = 1]) = P_{FA}$ is the false alarm rate of a pixel sensor. False alarms occur when the sensor falsely relates the presence of matter on its viewing line, when in fact there is none. $\mathcal{P}_f([\mathcal{F}_p^i = 0]) = 1 - P_{FA}$ is the rate with which we expect this pixel to correctly report non-detection.

We must assign a parametric form to $p(\mathcal{R} \,|\, \tau)$. There can be detection causes anywhere along the viewing line of p. We make no assumption about these causes and consider that detection is equally likely to be triggered by the voxel occupancy or by these causes. We therefore set this term to uniform. By doing this, we consider that accounting for the possibility itself is what is important, without necessarily giving an elaborate form to this term.

Parametric form for Sampling Term $p(\mathcal{S} \mid \tau)$. This term is dependent on i, p and X. We use uniform sampling, with $p(\mathcal{S} \mid \tau) = \mathcal{U}_{k\times k}(x - p)$. This gives equal weight to all voxels that fall within a $k \times k$ window around pixel p. A smoother, normal-based sampling could also be used but requires a higher computational cost to integrate information. Generally, the shape of this sampling function can easily be modified for specific needs. Both uniform and normal sampling forms enable some control over calibration, mis-synchronization, and some classification errors: several pixels will be able to contribute to a single voxel's decision upon inference. Thanks to the introduction of these two hidden processes and the given parametric forms, our method unifies broad silhouette uncertainty management and simple image sampling methods used in some visual hull algorithms such as [2]. It also enables to embed sub-voxel information about the underlying shape in the probability grid, as opposed to purely discrete approaches such as [14].

3.3. Image Formation Term

The image pixel likelihood term $p(\mathcal{I}_p^i \mid \mathcal{F}_p^i, \mathcal{B}_p^i, \tau)$ explains the color information of a pixel (i,p), given the knowledge of the background color and silhouette detection state at this pixel. We give two parametric forms to this term. If an object detection occurred at pixel (i,p), the knowledge about background images is irrelevant to the pixel's expected color: the background is known to be occluded by an object of interest, whose color the pixel observes. With no further assumption about colors of objects of interest, we consider them uniformly distributed: $p(\mathcal{I}_p^i \,|\, [\mathcal{F}_p^i = 1], \mathcal{B}_p^i, \tau) = \mathcal{U}(\mathcal{I}_p^i)$. Reciprocally, if no object detection occurred at this pixel, then the pixel's observed color should look similar to the pixel's background color. Such an expectancy can easily be formulated using a classical background model [17]:
$p(\mathcal{I}_p^i \mid [\mathcal{F}_p^i\!=\!0], [\mathcal{B}_p^i\!=\!(\mu_p^i, \sigma_p^i)], \tau) = \mathcal{N}(\mathcal{I}_p^i \mid \mu_p^i, \sigma_p^i)$, where (μ_p^i, σ_p^i) are the parameters of a Gaussian. The method could easily use any other background model, such as a

mixture of Gaussians [15], for sub-pixel noise robustness. Nevertheless, some problems persist whatever the background model: color ambiguities between foreground and background objects, lighting, or scene geometry change. It is the goal of our integrated multi-view approach to compensate for these weaknesses of single-view estimation.

4. Voxel Occupancy Inference

Once the joint probability distribution has been fully determined, it is possible to use Bayes' rule to infer the probability distributions of our *searched* variable $\mathcal{G}_X$, given the value of our *known* variables $\mathcal{I}, \mathcal{B}, \tau$, and marginalizing over *unknown* variables $\mathcal{F}$:

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

$$= \frac{\prod_{i,p} \sum_{\mathcal{F}_p^i} p(\mathcal{F}_p^i \,|\, \mathcal{G}_X, \tau) p(\mathcal{I}_p^i \,|\, \mathcal{F}_p^i, \mathcal{B}_p^i, \tau)}{\sum_{\mathcal{G}_X} \prod_{i,p} \sum_{\mathcal{F}_p^i} p(\mathcal{F}_p^i \,|\, \mathcal{G}_X, \tau) p(\mathcal{I}_p^i \,|\, \mathcal{F}_p^i, \mathcal{B}_p^i, \tau)} \quad (4)$$

after substitution of (1), and factorization. More details can be found in [6].

Note that the final inference expression (4) deceptively relates our voxel occupancy to *all* pixel observations. As we compute this inference *per voxel*, this is of course intractable. In practice, detection probabilities of pixels too far from the voxel's projection degenerate to uniform, as expressed in equations (2) and (3). Their contribution therefore factors out of the inference expression (4). The inference product can then be computed over a $k \times k$ window of pixels centered at the image projection of X, in each image. With a voxel grid size of N^3, the complexity of inferring all voxels of the grid is then $O(n\ k^2 N^3)$.

5. Results and Applications

We have implemented the proposed fusion approach, using uniform voxel sampling for experiments. Compared to normal sampling it is a good trade-off between computational cost and power of information integration. Notably the method has only three parameters $\{P_D, P_{FA}, k\}$, respectively the detection and false alarm rates, and the sampling window size, all of which can often be fixed for a given application. P_D and P_{FA} ponderate the confidence given to the observations. If $P_{FA}=0$ and $P_D=1$, then we trust observations blindly. If P_{FA} and P_D are close to 0.5 then observations are not trustworthy: it takes many more observations to conclude about the occupancy. k decides how broadly each image is sampled. We have tested the algorithm under various conditions, as it can be applied to many application fields. An associated video of results is available[1].

[1]http://movi.inrialpes.fr/Publications/2005/FB05/SilhouetteCueFusion.avi

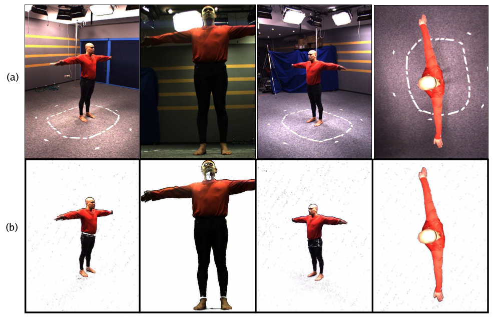


Figure 3. Inputs. (a) Four of the eight input images of the walking sequence (8 cameras, 15Hz acquisition) (b). Result given by monocular subtraction (semi-transparent rendering pondered by silhouette probability). Difficulties: camera 2 misses the subject's left forearm. Holes and noise appear in various silhouettes.

Modeling from Images. The grid itself is an estimate of shape. We illustrate this using the walking sequence. This sequence was acquired using 8 cameras of different characteristics (640×480, 780×580) at 15Hz. As Fig. 3 illustrates, the silhouette information that can be retrieved using monocular background subtraction is noisy. Also note that some cameras may not see the entire object during the sequence. These single-view subtractions also use a Gaussian background model, and reflect what input is available to our algorithm. Fig. 4 shows our method's results on a frame of the walking sequence, using a 120^3 grid. Cross-sections show how the shape information is embedded in the grid. See the associated video[1] for a dynamic view. As shown in Fig. 4(c), good surface modeling results can be achieved by extracting an isosurface from the probability grid. Fine, sub-voxel detail of the surface is recovered, and holes occurring in monocular subtractions are often filled. Additional modeling results are shown in Fig. 5.

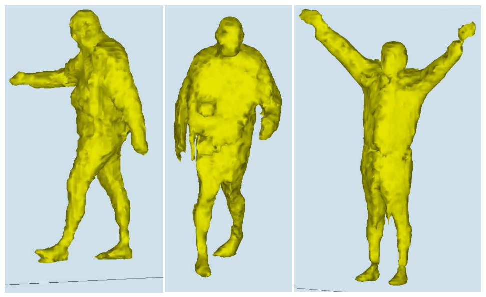


Figure 5. Isosurface of probability 0.80 at different time instants of the walking sequence. See video[1].

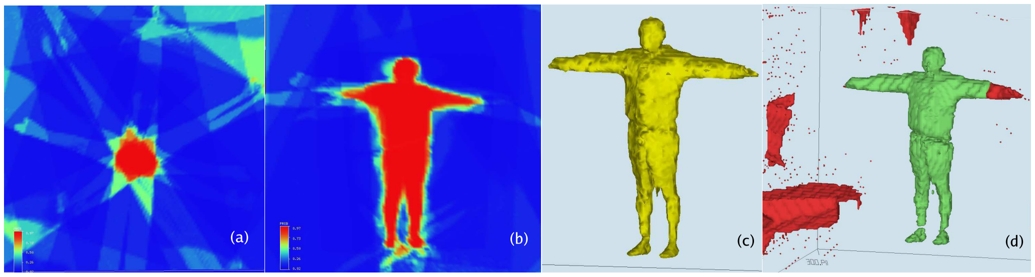


Figure 4. Walking sequence, acquired at 15Hz, using 8 cameras, with a 120^3 voxel grid. Computation time: approximately 13s on a 2.4 GHz PC. Parameters used: $P_D = 0.9, P_{FA} = 0.1, k = 5$ (a) Horizontal (chest) cross-section of the grid. Upper-left greenish regions are not seen by any camera (probability 0.5). (b) vertical grid cross-section. (c) Isosurface of probability 0.80 obtained from the grid. (d) Two classical visual hull reconstruction schemes: in light color, assuming common visibility of the object by all cameras. The forearm is lost. In dark, assuming that what is outside the visibility domain of a camera can be part of the visual hull. The latter recovers the left forearm, but ghost objects appear, in regions located in the visibility domain of a small number of cameras. Ghost objects appear when such regions project inside all silhouettes of views where they are seen.

The classical voxel-based visual hull approach has been implemented for comparison, with results in Fig. 4(d), where each voxel is carved if it projects outside silhouettes. We use the background subtractions of Fig. 3 for this experiment, and manually choose the best threshold *in each image* to provide binary silhouettes to the algorithm. Some holes are left unfilled by this method. Note that our method recovers valid occupancies from views that don't see the object entirely. This is transparent to the algorithm, because it only integrates information from sensors which see the voxels. This is unlike all classical, surface or volumetric visual hull approaches, where explicit assumptions are made about regions that project outside the visibility domain of an image, with various implications (see Fig. 4(d)).

Multi-View Background Subtraction. Our method computes a fusion of silhouette cues. This information can be used to compute consistent silhouettes in our input images, by re-projecting and rendering the occupancy grid from our input views, using a maximum intensity projection approach (MIP): for each pixel in an image, we collect the maximum probability in the grid along its viewing line. The goal is to express where silhouette detection is more likely in images. It would be possible to use the proposed statistical model to infer silhouette probability maps, given all image observations. This is however very expensive as it requires marginalization over voxel states, thus the proposed heuristic for multi-view background subtraction. All silhouettes can be extracted using a single threshold for all images. The advantage over monocular silhouette extraction is that each view benefits from the knowledge of silhouette information in the other views: resulting silhouettes show improvement, with fine details preserved (see Fig. 6). Small aliasing artifacts may appear depending on grid resolution and scene configuration.

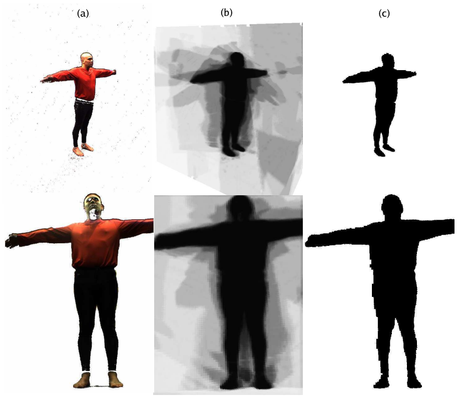


Figure 6. Multi-view silhouette extraction. (a) Monocular subtraction. Note the various artifacts and holes in such silhouettes (waist, head, feet). (b) MIP rendering of occupancy grid (120^3) probabilities from original viewpoints. Darker regions are more likely to be silhouette regions. (c) Thresholded version of (b), using a common threshold. Silhouettes show improvement. Unwanted dilatation only appears in concave regions, seen empty by a small number of cameras (crotch): the method outperforms most low-level monocular silhouette repairing schemes, such as morphological operations, for such large artifacts.

Object detection. The method can be used in much harder conditions to infer scene information. In the presence of high levels of noise, the size of the sampling window can be increased for additional robustness, with however a negative impact on precision (this tends to dilate the probability volume). Such noisy conditions limit the use of the method for 3D modeling and precise surface extraction; however the method can still be used reliably to *locate* objects in the scene. We illustrate this potential for object

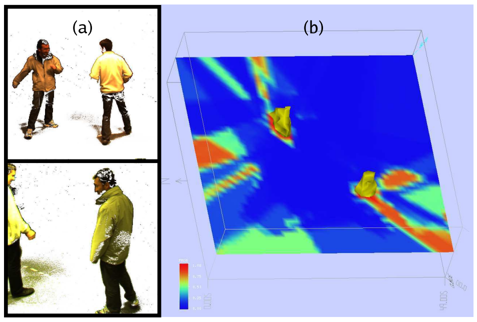


Figure 7. Multi-object sequence, with 8 cameras. (a) Difficult conditions yield very noisy single-view silhouette extractions. (b) Coarse grid ($50 \times 50 \times 18$) of the scene reconstructed with our method (computation time 7s), sufficient to localize objects (using $k = 25$). A horizontal cross-section of the grid, as well as the 0.67-probability isosurface, are shown (see the associated video). This is sufficient to localize the people.

localization, in an experiment with loose camera configurations and poor contrast images (see Fig. 7). 8 cameras are placed such that a relatively large area ($25m^2$) can be monitored in the room. Most cameras see the center of the room, but peripheral regions of this area are seen by 3 or 4 cameras at most. Two people walk randomly in the scene and are successfully localized, when seen by at least 3 cameras.

6. Discussion

We have presented a novel approach for silhouette cue fusion from multiple views. We use a rigorous sensor fusion framework, to relate scene information directly to observations. This has various advantages: the entire causality chain is modeled and all assumptions made explicit. It also avoids making hard decisions about silhouette labeling in images, which would have required tedious per-image parameter settings. Thus the underlying silhouette information in images can be smoothly integrated, using only three global parameters of a pixel sensor model. These parameters intuitively express the reliability of observations. This approach has been validated with several applications, and many new ideas can be experimented and plugged-in without changing the core of the method.

Arguably, more dependencies could be considered in the model. Namely, we notice that the reliability of pixel decision can be related to the colors observed at this pixel: many times we observe the case where black foreground objects are observed in front of a black background and misclassified, a case which could be explicitly modeled. The local nature of grid evaluations opens the possibility for a real-time, hardware-accelerated solution. More generally, our model estimates static grids at one time instant. It would greatly benefit from temporal consistency, where passed observations are used to infer current occupancy states. Happily, occupancy grids provide a good framework for temporal accumulation of information, being one of its main uses in the robotics community [3]. We will investigate these possibilities to extend the capabilities of our system.

References

[1] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for the Space Carving algorithm. In *ICCV'01*, pages 388–393, 2001.
[2] K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *CVPR'00*, volume 2, pages 714 – 720, 2000.
[3] A. Elfes. *Occupancy grids: a probabilistic framework for robot perception and navigation*. PhD thesis, 1989.
[4] O. Faugeras and R. Keriven. Complete dense stereovision using level set methods. In *ECCV'98*, volume I, pages 379+, 1998.
[5] J. S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In *BMVC'03*, 2003.
[6] J. S. Franco and E. Boyer. Fusion of Multi-View Silhouette Cues Using an Occupancy Grid. Technical report, INRIA RR-5551, Apr. 2005.
[7] B. Goldlücke and M. Magnor. Joint 3-d reconstruction and background separation in multiple views using graph cuts. *CVPR'03*, I:683–694, 2003.
[8] K. Grauman, G. Shakhnarovich, and T. Darrell. A bayesian approach to image-based visual hull reconstruction. In *CVPR'03*, pages 187–194, 2003.
[9] J. Isidoro and S. Sclaroff. Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints. In *ICCV'03*, pages 1335–1342, 2003.
[10] K. Kutulakos and S. Seitz. A Theory of Shape by Space Carving. *IJCV*, 38(3):199–218, 2000.
[11] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2):150–162, Feb. 1994.
[12] 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.
[13] 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.
[14] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *CVPR'00*, pages 345–353, 2000.
[15] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR'99*, pages 2246–2252, 1999.
[16] R. Szeliski. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 58(1):23–32, 1993.
[17] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on PAMI*, 19(7):780–785, 1997.
[18] G. Zeng and L. Quan. Silhouette extraction from multiple images of an unknown background. In *ACCV'04*, 2004.

6.7 A Distributed Approach for Real-Time 3D Modeling

Jean-Sébastien Franco, Clément Ménier, Edmond Boyer and Bruno Raffin

Proceedings of the IEEE CVPR Workshop on Real-Time 3D Sensors and their Applications,

Washington (USA), June 2004

A Distributed Approach for Real Time 3D Modeling

Jean-Sébastien Franco[1] Clément Ménier[1,2] Edmond Boyer[1] Bruno Raffin[2]
[1]GRAVIR - INRIA Rhône-Alpes, France [2]ID - INRIA Rhône-Alpes, France

Abstract

This paper addresses the problem of real time 3D modeling from images with multiple cameras. Environments where multiple cameras and PCs are present are becoming usual, mainly due to new camera technologies and high computing power of modern PCs. However most applications in computer vision are based on a single, or few PCs for computations and do not scale. Our motivation in this paper is therefore to propose a distributed framework which allows to compute precise 3D models in real time with a variable number of cameras, this through an optimal use of the several PCs which are generally present. We focus in this paper on silhouette based modeling approaches and investigate how to efficiently partition the associated tasks over a set of PCs. Our contribution is a distribution scheme that applies to the different types of approaches in this field and allows for real time applications. Such a scheme relies on different accessible levels of parallelization, from individual task partitions to concurrent executions, yielding in turn controls on both latency and frame rate of the modeling system. We report on the application of the presented framework to visual hull modeling applications. In particular, we show that precise surface models can be computed in real time with standard components. Results with synthetic data and preliminary results in real contexts are presented.

1. Introduction

Recent advances in camera technologies have generalized real time acquisition of digital images, and today, any modern PC can acquire such images at standard video rates. This allows complete acquisition systems to be built by simply connecting sets of digital cameras and PCs, without help from specific components. The interest arises in various application domains where digital cameras are involved and where image information extracted in real time is required for interaction or control purposes. These domains include, for instance, scene virtualizations, video surveillances or human-machine interfaces. However, while much research has been devoted to algorithms that address the situation where a few cameras are connected to a single or few PCs, less efforts have been made toward larger sets of cameras and PCs which are, on the other hand, becoming standard. Moreover, most computer vision applications that make use of several cameras connected to several PCs do not take advantage of the available computing power and generally rely on a single PC for computations. In this paper, we address these scalability and optimality issues by considering parallel strategies for 3D modeling computations. The objective is to propose practical and scalable solutions to produce highly precise 3D models in real time using multiple cameras.

Only a few distributed 3D modeling systems have been proposed and demonstrated in the past. The CMU robotics institute introduced a 3D dome with around 50 cameras for virtualization [Narayanan98]; the 3D scene model being built using a stereo-vision approach. Other systems have also been proposed at CMU with fewer cameras and a voxel based approach [Cheung00], or its combination with a stereo based approach [Cheung03]. However, these systems do either work off-line, or in real time but with a few cameras, since no parallelisation is considered for modeling computations. Another class of real time but non-parallel approaches make use of graphic cards to directly render new viewpoint images [Li03]. Using graphic card hardware highly accelerates the rendering process but such systems still rely on a single PC for computations and do not provide explicit 3D models as required by many applications. In [Borovikov03], a parallel framework that stores, retrieves and processes video streams on PC clusters is presented, but the emphasis is put on data management and real time applications are not considered. Real time and parallel modeling systems have also been proposed to handle voxel based modeling methods [Kameda00, Arita01]. Voxel based methods produce discrete 3D models by using regular space discretizations where parallelepipedic cells -the voxels- are carved according to image information [Slabaugh01, Dyer01]. Such methods can easily be parallelized since a significant part of the computations is achieved independently per voxel. The mentioned approaches make use of this fact and report good performance. However, voxel based methods are imprecise and time consuming. Furthermore the principles developed in the proposed approaches do not easily extend to possibly better modeling methods. In this paper, we attempt to develop concepts at a higher abstraction level in order to make their application possible to various modeling methods. Another interesting work [François01] deals with video streams and proposes a multi-threading strategy to improve latency and frame rate when processing video streams for segmentation or tracking tasks. We mention also here the Skipper project [Sérot02] which develops a parallel programming environment for image processing. The two latter works also tackle distribution issues related to computer vision, but they provide middleware solutions mainly and do not consider algorithmic aspects as necessary with the targeted modeling applications.

In this paper, we present a real time and parallel architecture for multi-view algorithms and report on its application to 3D modeling applications and, in particular, to silhouette based approaches. Our goal is to provide scalable solutions by using a parallelization methodology. Such a methodology is intended to be general enough to apply to different contexts while maintaining the required parallelization effort reasonable. Our contribution with respect to the mentioned works is twofold: first, we extend parallelization concepts already proposed to multi-view algorithms; second, we apply this concept to silhouette based approaches and propose new practical implementations that are scalable while surpassing standard voxel based approaches.

The paper is organized as follows. Section 2 introduces our distribution scheme for parallelizing multi-view tasks. In section 3, their application to visual hull computations using image silhouettes is presented. Section 4 demonstrates the proposed principles in a real context and gives numerical evidence on synthetic data.

2. Distribution Scheme

In this section we present our methodology to parallelize multi-view algorithms. It relies on classical parallelization techniques. The simplicity of these techniques limit the effort required to parallelize existing algorithms. Experimental results show that this methodology leads to good performance.

Let n be the number of cameras available and m the number of hosts (PCs). We assume each camera is connected to one host dedicated to acquisition and local image processing. We consider that all cameras issue images at the same *frame rate*. We will consider that a *frame* corresponds to the set of n images taken at a time t. We also assume that m is greater than n. The extra $p = m-n$ hosts are dedicated to computation. Hosts are interconnected through a standard network. Accessing data on a distant host takes much more time than accessing local data. We do not use any tool masking this disparity, like a virtually shared memory. They generally lead to lower performance than tools that enable the user to take into account this disparity to optimize data transfers, like message passing libraries [Gropp94]. As we will see, the effort of explicitly handling data transfers is relatively low and worthwhile.

As a performance criterion, we measure the *speed-up*, obtained by dividing the sequential execution time by the parallel execution time. The efficiency of the parallelization increases as the speed-up factor nears the number of processors. For real-time constraints, we measure the frame processing rate and the *latency*, i.e. the time to process a single frame.

The methodology we propose is based on two different levels of parallelization, a *stream level* parallelization and a *frame level* parallelization.

2.1. Stream Level Parallelization

Multi-view applications process a sequence of frames, called a *stream*. A very classical method to speed up stream based applications is to use a *pipeline*. The application is split in a sequence of stages (see fig. 1(b)), each stage being executed on different hosts. It enables a host to work on frame t while the next host in the pipe-line stage works on frame $t-1$. The different stages are naturally extracted from the structure of the program. Trying to redesign the application into a longer pipe-line is time consuming and increases the latency due to extra communications.

A stage is *time-independent* if it does not use temporal consistency, i.e. the process of frame t does not depend on the results from preceding frames. It enables to duplicate identical frame computation schemes on different hosts, called *processing units* (see fig. 1(c)). A new frame t can be processed as soon as one of the processing units is available. The number of processing units should be large enough to avoid any frame to wait for its processing. Adapting this technique to a time-dependent stage may still be possible but requires advanced scheduling strategies and extra communications.

This scheme can be applied to the classical voxel-based approach. Frames go through 2 processing steps, background extraction and voxel carving that can be assigned to 2 pipe-line stages. The first stage being usually much faster than the second one, several processing units can be dedicated to this time-independent second stage.

2.2. Frame Level Parallelization

The preceding distribution techniques can significantly improve the processing frame rate. However, the latency is negatively affected. The pipe-line introduces extra communication time that increases the latency, thus reducing the reactivity of the system. To improve latency, one can re-

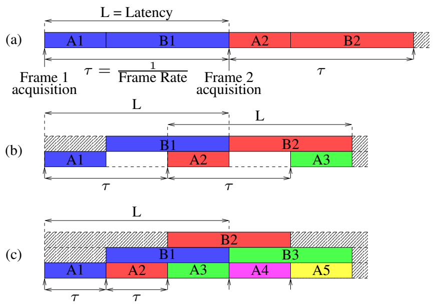


Figure 1: Different levels of parallelization proposed by our framework. A and B are the two computation stages of the program. At and Bt relate to the processing of frame t. Each row corresponds to a processor. Colored blocks correspond to a task execution and white blocks to inactivity periods. The graph (a) represents a sequential execution, the graph (b) a 2 stage pipeline execution. Graph (c) adds a second processing unit for the second pipeline stage B.

duce the time taken by a stage to process a single frame. This can be done by parallelizing the work done on a single frame among several hosts of a single processing unit (see fig. 2). We base our scheme on the classical *Bulk Synchronous Programming* (BSP) model [Valiant90] that proposes a trade-off between performance and programming complexity. The execution is done in a sequence of phases, each one decomposed in a data exchange involving all hosts followed by local computations performed on each host. This model eases parallel algorithm description as it splits communication from computation. Based on this BSP approach, we propose a 3 phase scheme for parallelizing processing unit computations:

- **Data preparation**: the first phase (input data distribution) consists in sending the input data to the hosts requiring them. Next, each host locally performs the initialization computations needed for the next parallel phase.
- **Parallel computation**: in parallel, each host executes locally (no communication) a different task assigned to it.
- **Sequential computation**: all partial results from the parallel computation phase are gathered on one host. This host sequentially performs the remaining computation that could not have been parallelized in the previous phase. Depending on the requirements of the next pipe-line stage, sequential computation can be duplicated on several hosts. It can enable to reduce communication load to transfer data to this next stage.

Though very simple this model is very generic. In worst cases, all the computation is done in the last sequential

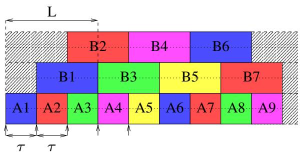


Figure 2: Frame level parallelization (2 processors for stage A and 4 processors for stage B). It shows latency improvement in comparison with stream level parallelization (see fig. 1.)

phase. However this is trivially inefficient. We later show that it is possible to obtain a parallel computation phase significantly larger than the two other phases. In such a situation we will show that this scheme leads to real-time performance.

We can illustrate application of our scheme to the parallelization of voxel carving as described by Arita *et al.* [Arita01]:

- Data preparation: initialize locally the voxel space.
- Parallel computation: for each image, compute the visual cone in the local voxel space.
- Sequential computation: gather all visual cones on one host and compute their intersection.

Results given by Arita show that this scheme yields high performance. Generally, our 3 phase scheme does not lead to an optimal parallelization. Many optimizations can still be done. On this voxel example, Borovikov *et al.* has given an algorithmic optimization [Borovikov03] for computing the voxel cone intersection in a more complex way that cannot be represented with this 3 phase scheme. However we show in this paper that the proposed methodology offers a simple yet efficient scheme to address stream and frame level parallelization for real-time constraints.

3. Silhouette-Based Modeling Approaches

We now deal with silhouette-based modeling approaches and how to make them suitable for a real-time context using our methodology. We focus on visual hull reconstruction approaches, as they are quite popularly used for 3D modeling from multiple views given their speed and simplicity. Recall that the visual hull is a simple approximation of the scene objects defined as the maximal shape consistent with the silhouettes of the objects in the input views [Laurentini94]. A number of algorithms have been proposed for computing the visual hull. Some use a discrete partitioning of space in voxels. Such volume-based

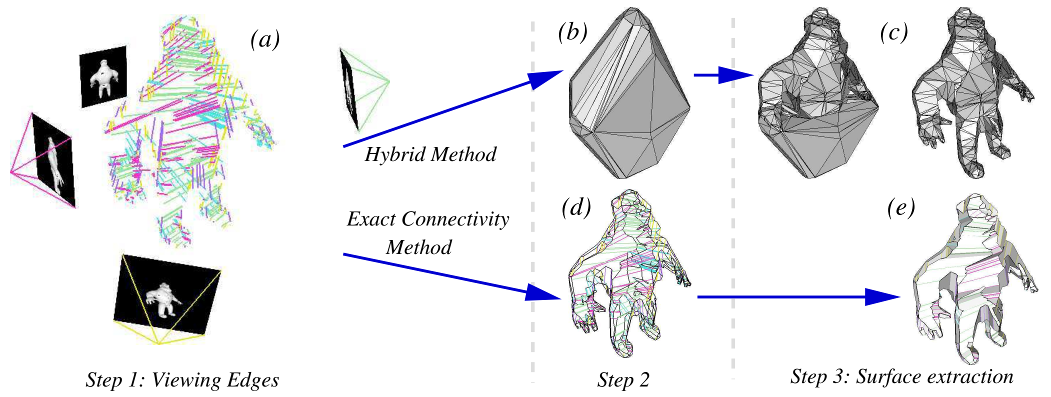


Figure 3: Outline of the visual hull modeling techniques chosen for parallelization. (a) Images of an object are taken, silhouettes are identified, their contours vectorized, and viewing edges are computed for each point of the discretization. (b) The hybrid method computes the Delaunay tetrahedron decomposition of space based on viewing edge vertices. (c) Each tetrahedron is carved according to silhouette consistency, and the final visual hull model is obtained. (d) The exact connectivity method computes the cone intersection components belonging to the visual hull, to complete the entire visual hull polyhedron mesh. (e) Faces are extracted from the mesh representation and the final polyhedron model of the visual hull is obtained.

schemes prove to be simple to implement and can be very fast. We have given a case study for the parallelization of these approaches in the previous section. In this section, we will therefore focus on the parallelization of methods that have never been studied before in this context.

Namely, a recently popular category of *surface-based* modeling approaches have focused on recovering the surface of the visual hull, and provide a polyhedral representation of the visual hull surface as output [Baumgart75, Matusik01], some giving additional topological guarantees with a simpler framework [Franco03]. An interesting hybrid approach also exists that combines advantages of both volume and surface-based families [Boyer03]. While these methods provide a precise model of the visual hull and are generally fast, they are still too slow for a hard real-time setup with as many as 10 cameras. On the other hand this makes them outstanding potential beneficiaries of parallelization. In this context we can show that parallelism is a tool to bridge the gap between generally fast vision algorithms, and vision algorithms that *guarantee* very high frame processing rates of 30fps or above.

3.1. Outline of the Modeling Methods

In order to offer a broad view of the parallelization of silhouette-based approaches, we will focus on two of the most recent methods, the *hybrid* method [Boyer03], which offers a robust trade-off between volume and surface-based approaches, and one of the available surface-based methods, the *exact connectivity* method [Franco03]. See figure 3 for an overview. Recall the context of such methods: n calibrated cameras are used to generate n views of an object; a standard background subtraction process is used to extract the silhouette bitmaps from the images. The contours of the obtained silhouette masks are vectorized so as to obtain oriented 2D polygons bounding the silhouette regions. This discrete representation of silhouettes induces a discrete visual hull polyhedron. The hybrid method provides a close approximation of this polyhedron, while the exact connectivity method computes it exactly.

Three steps are used to achieve the reconstruction goal in both cases, as depicted in figure 3. The first step, common to both methods, computes an initial subset of the visual hull geometry, the *viewing edges*, in the form of points and edges located on the viewing lines of each discrete silhouette contour point (details follow in 3.2). The second step's common goal is to compute an intermediate representation which implicitely contains the visual hull surface. To this goal, the hybrid method partitions space into convex cells, which can easily be carved according to silhouette consistency of their projection in images. In contrast, the exact connectivity method computes the exact visual hull polyhedron as a generalized cone intersection. Finally, the third step's common goal is to identify the underlying surface information, by extracting the visual hull interface polygons from the previous representation. The following sections give more details about these steps.

Note that, as a first possibility for applying our methodology, we can easily identify each conceptual step of the methods with a stage in a multi processing unit pipe-line, to increase the output frame rate. This is valid since the algorithms are intrisincally time-independent: each set of silhouettes in frame t is used for reconstructing a shape, and the information will never be used again in subsequent frames. We will now deal with more specific issues in the following sections.

3.2. Computing the Viewing Edges

We now describe the computation of viewing edges at discrete image contour vertices, as it is a common processing step in the presented methods (see fig. 3).

Viewing edges are intervals along viewing lines. They correspond to viewing lines contributions to the visual hull surface and are thus associated to image points on silhouette contours. As such, viewing edges are simply obtained by computing the set of intervals along a viewing line that project inside all silhouettes (see fig. 4).

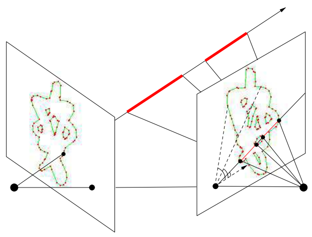


Figure 4: Viewing edges (in bold) along the viewing line. Epipolar line angles can be used to accelerate the search for the image segments intersecting the epipolar line.

Interestingly, this algorithm provides great freedom for frame-level parallelism, as it consists in the computation of a set of numerous partial but independent results. That is, each viewing line's contributions can be computed regardless of all others, assuming the silhouette information from all images is available. An efficient frame-level parallelization of viewing edge computation can hence be obtained by partitioning all viewing lines of all images in p sets during the data preparation phase, and distributing each batch to one of p hosts for processing (parallel computation phase). One must be careful in balancing the workload between hosts, in order to reduce the time spent waiting for the slowest host. Building sets of identical cardinality during data preparation proved to be efficient as we will show. Observe that this parallel scheme heavily constrains how we will perform data preparation: as each host requires all silhouette information, silhouette contours must also be broadcasted during that phase. Finalization of the task simply consists in gathering the union of all partial results on the hosts that require it, during the sequential computation phase.

We are able to achieve speed-ups of the order of 8 with 10 hosts, which is very good, especially given the low effort required to parallelize the algorithm. Higher speed-ups can be achieved, but with a substantially higher complexity, much at the expense of the gain/effort tradeoff.

3.3. A Distributed Hybrid Method

We will now describe the parallelization of the hybrid method [Boyer03]. After computing the viewing edges, the hybrid method uses a Delaunay triangulation of the viewing edge vertices to obtain a decomposition of space into tetrahedrons, as a second step (see fig. 3). The union of these tetrahedrons form the convex hull of the input points: some of them must be carved in order to isolate the visual hull. The discretization consists of convex cells of a more generalized and flexible shape than regular voxels, but can still be carved with voxel-like silhouette consistency checks such as those in [Cheung00]. This is used in the third step to determine which tetrahedrons lie inside or outside the visual hull, and the surface polygons are extracted from this model by simply isolating the triangles which lie at the interface between the two regions.

Although the algorithm is conceptually simple, building its parallel counterpart brings challenges we have to account for as we seek to apply our proposed methodology. The main issue here is the Delaunay triangulation, which generates many partial, but globally constrained results: the Delaunay tetrahedrons. Some possibilities for distributing the Delaunay triangulation have been explored [Cignoni93], with the main idea of subdividing the problem among space regions where concurrent tasks take place. This idea can be applied to many vision algorithms. One obstacle, also widely generic, is the complexity of detecting and dealing with region interrelationships in the algorithm. In the case of the Delaunay triangulation, a programmer would spend most of his time re-implementing the tedious algorithmics intrinsic to such a method. Under such conditions, it is wise to sacrifice system reactivity to implementation simplicity. Stream level parallelization can still be used to improve the frame rate of an already available sequential code. Yet we will see in the next section a case where tackling the multiple region interdependency problem is worthwhile.

However we do have another opportunity for parallelization, as the cell carving task is much friendlier. Much like in a usual volume-based technique, the per-cell carving results are independent, which ensures a well-behaved parallel execution phase. The only requirement is that all silhouette information is available at all hosts, which can be provided for during data preparation. The sequential execution phase will then simply gather the carve state of all tetrahedrons, and finally extract the surface triangles from this information, as this takes very little time and does not require distribution. Under such favorable conditions we are able in this carving task context to reach speed-ups of 9.5 for 10 hosts.

3.4. A Distributed Surface-Based Method

We now briefly describe the application of our proposed parallelization methodology on the exact connectiv-

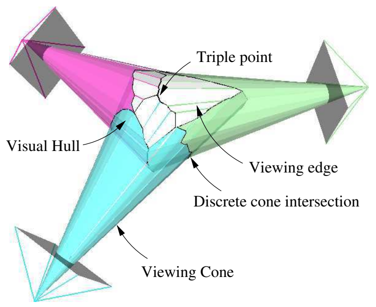


Figure 5: Visual hull of a sphere with 3 views.

ity method [Franco03] (overview available in figure 3). Figure 5 provides a representation of all geometric entities involved.

As seen previously, the viewing edges give us an initial subset of the visual hull geometry. However, this does not account for all edges of the visual hull polyhedron. The visual hull is the intersection of the *viewing cones*, which back-project from silhouettes in images. For a complete visual hull, one must also compute the missing cone intersection curves that participate to the visual hull polyhedron. It can be observed that such curves, which are piecewise-linear, snake around the visual hull surface, connecting together viewing edge vertices and extra vertices called *triple points*. Triple points are the locus of three cone intersections; as such they are always the meeting point of three cone intersection curves on the visual hull (see fig. 5).

With this in mind, the exact method simply seeks to follow these curves while creating them, starting from viewing edge vertices and recursing at cone intersection junctions, i.e. the triple points. The algorithm does so by iteratively computing new edges of the curve, using incidence information previously inferred. Checking for the silhouette consistency of each newly created edge ensures that it is part of the visual hull mesh; it also enables the algorithm to detect and create the missing triple points, when such consistency is violated. When all cone intersection edges are recovered, faces of the polyhedron surface are extracted by walking through the complete oriented mesh while always taking left turns at each vertex, so as to identify each face's 2D contours and complete the third step in the algorithm. Refer to [Franco03] for details.

Any parallelization effort for this algorithm will likely be confronted to the strong spatial dependencies inherent to a mesh representation. In order to allow for concurrent task execution, we classically choose to partition space into p different regions using $p-1$ parallel planes, thus subdividing space in p "slices". Slice width is adjusted by attributing a constant number of viewing edge vertices per slice for workload balancing. Since we mainly manipulate edges and vertices, partitioning of primitives among regions during the data preparation phase is a very low cost operation. Thus, a host of the distributed exact connectivity method can be instructed to follow intersection curves within its dedicated region $\mathcal{R}_i$, until this curve crosses the border toward another region $\mathcal{R}_j$. The host then stops processing this curve, thereby delegating the computation of the rest of the curve to the host in charge of $\mathcal{R}_j$ during the parallel computation phase.

Observe that region dependencies are very easy to identify as they only materialize at edges that cross a partition plane. It is yet again straightforward to identify the three simple phases of our frame-level parallel model in this case. Data preparation partitions the job among regions; parallel computation tasks compute mesh fragments associated to their dedicated region; the sequential computation phase gathers and carefully merges the partial meshes across region borders. This proves to be very efficient as we reach speed-ups of 6 with 10 hosts with our implementation, an excellent result given the reasonable implementation time and the dependency issues. This will be confirmed by the global measurements provided in the next section.

We are also able to distribute the surface extraction step: the complete mesh is broadcasted to p hosts during data preparation, then the p hosts independently compute a subset of the face information, and the sequential finalization simply gathers all sets of faces. This leads to very good speed-ups of the order of 7 for 10 hosts.

4. Implementation and Experimental Results

In this section, we detail experimental results obtained from the implementation of the two preceding algorithms paralellized with our methodology. We obtain real time performance for high quality 3D modeling; recall that for the second method the computed polyhedron is exact with respect to the input silhouettes. Tests with synthetic data show that sustained performance is obtained with a large number of view points.

Our 16 processor cluster is composed of 8 dual Xeon PCs (2.66 GHz) connected through a Gigabit Ethernet network. Latency is measured from the beginning of the viewing-edge step. Our implementation uses the standard MPI message passing library [Gropp94] for comunications. The Delaunay triangulation is computed with the high performance sequential library Qhull [Qhu]. All presented results are based on sets of 10 experiments.

4.1. Real Time Conditions

Our real experimental setup is composed of 4 IEEE 1394 cameras each connected to a PC handling acquisition, background substraction and silhouette vectorization. Images (640x480) are acquired at 30 frames per second. The scene is composed of a person (see fig. 6), an object of average complexity (150 contour vertices per image).

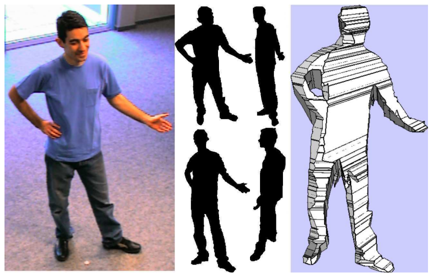


Figure 6: Real time reconstruction of a real person with 4 cameras and the exact connectivity approach.

Using such a system to run the hybrid method, we achieve real time 3D modeling at 30 frames per second using 16 processors. One processing unit parallelized on 4 hosts is dedicated to the first step. Ten processing units are dedicated to the sequential Delaunay triangulation. Carving is achieved in a single processing unit parallelized on 2 processors. The measured latency comes in the average of 400 ms, but is highly limited by the sequential execution time of the triangulation time, which can reach 300 ms.

The exact connectivity method proved to be more efficient as real time execution (30 frames per second) is achieved with only 12 processors. Each stage has 2 processing units, each one being parallelized on 2 processors. The measured latency is about 100 ms. This low latency and real time frame processing rate enable to use this algorithm for interactive applications. Videos are available at `http://www.inrialpes.fr/movi/people/Franco/CVPR04`.

4.2. Validation with Large Numbers of View Points

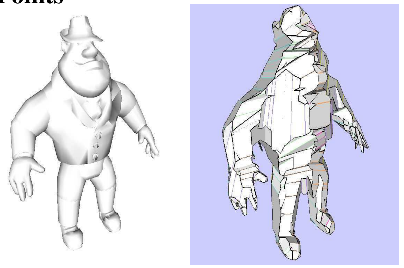


Figure 7: (left) Original model. (right) Reconstruction of the Model with 12 view points.

Not having more than 4 cameras available, the scalability of our distribued algorithms was tested with images from multiple view points generated from a synthetic model. We focus on the latency issue. We only consider the exact connectivity 3D modeling algorithm as the hybrid one is latency limited by the Delaunay triangulation. The real time frame rate issue is not discussed as it can be solved by multiplying the number of PCs assigned to the stream level parallelization.

The model we consider is a synthetic person with a complexity close to a real person (about 130 contour vertices per image). Figure 8 presents the obtained latency with regard to the number of processors involved for 16, 25 and 64 view points. The parallelization of the algorithm enables to significantly reduce the latency (almost by an order of magnitude). With 25 view points and 16 processors, latency is below 200 ms, a latency level suitable for interactivity.

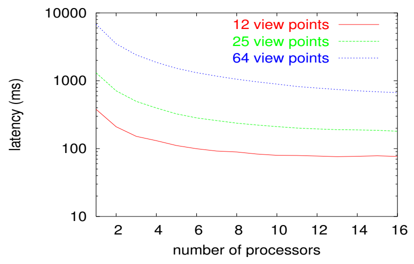


Figure 8: Log plot latencies for the synthetic person.

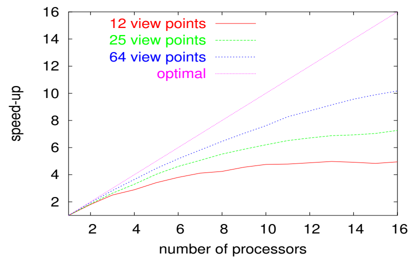


Figure 9: Speed-ups for the synthetic person.

Figure 9 presents the associated speed-ups. Up to 9 processors for 12 view points, 14 processors for 25 view points, and more than 16 processors for 64 view points, the speed-up is above half of the processors used. Next, the speed-ups tend to stabilize as the workload in the parallel computation phases decreases compared to the data preparation and sequential computation phases.

5. Summary and Conclusions

We have presented a 3D modeling system which uses parallelism to reach real time executions with a flexible number of cameras and PCs. Such system is based on a distribution framework we propose, which is intended to multi-view applications in computer vision. We have demonstrated its effectiveness in 3D modeling applications using silhouettes. The high quality visual hulls generated by these parallel algorithms can be used for various applications, including virtual reality (see fig. 10). Our main contribution with respect to existing works in the field is to provide new parallel 3D modeling implementations as well as a methodology for the parallelization of multi-view applications. Results on real and synthetic data show that our approach allows for scalability in modeling systems and extends therefore the potential of such systems. We are currently studying generalization of the given principles to other computer vision applications. We are also extending our experimental setup so that it includes more than 20 cameras and provides a complete pipe-line from the image acquisition to the model visualization in multi-projector environments, with all the associated tasks distributed on a PC cluster.

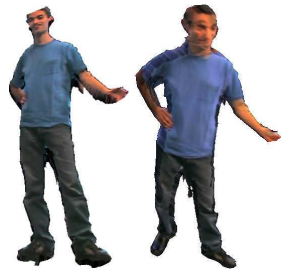


Figure 10: Two new views of the visual hull model of figure 6 with view-dependent texturing.

References

[Arita01] D. Arita and R.-I. Taniguchi. RPV-II: A Stream-Based Real-Time Parallel Vision System and Its Application to Real-Time Volume Reconstruction. In *Computer Vision Systems, Second International Workshop, ICVS, Vancouver (Canada)*, 2001.

[Baumgart75] B.G. Baumgart. A polyhedron representation for computer vision. In *AFIPS National Computer Conference*, 1975.

[Borovikov03] 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.

[Boyer03] E. Boyer and J.-S. Franco. A Hybrid Approach for Computing Visual Hulls of Complex Objects. In *CVPR'2003, Madison (USA)*, volume I, pages 695–701, 2003.

[Cheung00] G. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *CVPR'2000, Hilton Head Island, (USA)*, volume 2, pages 714 – 720, June 2000.

[Cheung03] 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 *CVPR'2003, Madison (USA)*, 2003.

[Cignoni93] P. Cignoni, C. Montani, R. Perego, and R. Scopigno. Parallel 3D Delaunay Triangulation. *Computer Graphics Forum*, 12(3): 129–142, 1993.

[Dyer01] C.R. Dyer. Volumetric Scene Reconstruction from Multiple Views. In L.S. Davis, editor, *Foundations of Image Understanding*, pages 469–489. Kluwer, Boston, 2001.

[François01] A. François and G. Médioni. A Modular Software Architecture for Real Time Video Processing. In *Computer Vision Systems, Second International Workshop, ICVS, Vancouver (Canada)*, pages 35–49, 2001.

[Franco03] J.S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In *BMVC'2003, Norwich (UK)*, 2003.

[Gropp94] 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.

[Kameda00] 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.

[Laurentini94] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2): 150–162, February 1994.

[Li03] M. Li, M. Magnor, and H.-P. Seidel. Improved hardware-accelerated visual hull rendering. In *Vision, Modeling and Visualization Workshop, Munich, (Germany)*, 2003.

[Matusik01] W. Matusik, C. Buehler, and L. McMillan. Polyhedral Visual Hulls for Real-Time Rendering. In *Eurographics Workshop on Rendering*, 2001.

[Narayanan98] P.J. Narayanan, P.W. Rander, and T. Kanade. Constructing Virtual Wolrds Using Dense Stereo. In *ICCV'1998, Bombay, (India)*, pages 3–10, 1998.

[Qhu] Qhull, convex hull and delaunay triangulation library. http://www.geom.uiuc.edu/software/qhull/.

[Sérot02] J. Sérot and D. Ginhac. Skeletons for parallel image processing: an overview of the skipper project. *Parallel Computing*, 28(12): 1685–1708, 2002.

[Slabaugh01] 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.

[Valiant90] L. G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8): 103–111, August 1990.

6.8 The Grimage Platform : A Mixed Reality Environment for Interactions

Jérémie Allard, Jean-Sébastien Franco, Clément Ménier, Edmond Boyer and Bruno Raffin

Proceedings of the 4th International Conference on Computer Vision Systems

New York (USA), January 2006

The GrImage Platform: A Mixed Reality Environment for Interactions

Jérémie Allard Jean-Sébastien Franco Clément Ménier Edmond Boyer
Bruno Raffin

Laboratoire Gravir, Laboratoire ID
CNRS/INPG/INRIA/UJF
INRIA Rhône-Alpes
655 avenue de l'Europe, 38334 Saint Ismier, France

E-mail: `firstname.lastname@inrialpes.fr`

Abstract

In this paper, we present a scalable architecture to compute, visualize and interact with 3D dynamic models of real scenes. This architecture is designed for mixed reality applications requiring such dynamic models, tele-immersion for instance. Our system consists in 3 *main parts: the acquisition, based on standard firewire cameras; the computation, based on a distribution scheme over a cluster of PC and using a recent shape-from-silhouette algorithm which leads to optimally precise 3D models; the visualization, which is achieved on a multiple display wall. The proposed distribution scheme ensures scalability of the system and hereby allows control over the number of cameras used for acquisition, the frame-rate, or the number of projectors used for high resolution visualization. To our knowledge this is the first completely scalable vision architecture for real time 3D modeling, from acquisition to visualization through computation. Experimental results show that this framework is very promising for real time 3D interactions.*

1 Introduction

Interactive and mixed reality environments generally rely on the ability to retrieve 3D information about users, in real time, in an interaction space. Such information is used to make real and virtual worlds consistent with one another. Traditional solutions to this problem usually consist in tracking positions of sensors by means of various technologies including electromagnetic waves, infrared sensors or accelerometers. However, this requires users to wear invasive equipment and usually specific body suits. Furthermore it does not lead to a shape description, as required for many applications such as tele-immersion for example. In this paper, we consider a more flexible class of methods based on digital cameras. These methods can compute 3D shape models in real-time, and without any markers or any specific equipment. We propose a framework in this context, from acquisition to visualization and interactions. Our objective is to provide a flexible solution which especially focuses on issues that are critical in such systems: precision of the 3D model, precision of the visualization and process speed.

Several multi-camera systems for dynamic modeling have been proposed. Stereo based systems were first proposed [16] for *virtualization*, but most recent systems use image silhouettes as input data to compute 3D shapes. They can be classified according to the fact that they work offline or in real-time, and also by the type of 3D models which they build. Offline systems allow complex and precise models to be built [6, 5], in particular articulated models, however they do not allow real-time interaction as intended in this work. Most real-time systems, such as [7, 10], that have been proposed in the past, compute voxel models, i.e. discrete 3D models made of elementary parallelepipedic cells. Interestingly, several systems in this category [4, 12, 3, 18] use a distribution scheme over a PC cluster to speed up computations and hence, provide some kind of control over the model precision and the process speed. However, voxel based methods are still imprecise unless a huge number of voxels is used. Furthermore they require post-processing, typically a marching cubes approach, to produce surface shapes. This is computationally expensive, and generates very small-scale geometry whenever precision is required.

Another class of real time, but non-parallel, approaches directly render new viewpoint images [17] using possibly graphic cards for computations[14]. Based on the *Image Based Visual Hull* method [15], these approaches efficiently focus on the desired 2D image, but they still rely on a single PC for computations, limiting the number of video-streams or the frame-rate, and they do not provide explicit 3D shapes as required by many applications.

In contrast to the aforementioned systems, ours directly computes watertight and manifold surface models. These surface models are exact with respect to the input silhouette information available and, as such, are optimal and equivalent to voxel grids with infinite resolutions. A particular emphasis has been put on the system scalability to ensure flexibility and to address performance and hardware cost efficiency issues. To this aim, the system is composed of multiple commodity components: FireWire cameras distributed on multiple PCs interconnected through a standard Ethernet network, as well as multiple projectors for a wall display. To reach real time performance, a careful distribution of the work load on the different resources is achieved. For that purpose we rely on a middleware library called FlowVR [1], dedicated to the distribution of interactive applications.

Section 2 outlines the global approach. Section 3 discusses issues related to image acquisition. The 3D modeling algorithm and its parallel implementation is then explained in section 4. In section 5, interactions and visualization are described. Section 6 details the distributed framework for our system. Section 7 presents some experimental results before concluding in section 8.

2 Outline

Our goal is to compute 3D shapes of users in an acquisition space surrounded by several cameras in real time (see figure 1). Such models are subsequently used for interaction purposes, including display. In order to achieve this, several processes must be coupled.

Acquisition Fixed cameras are set to surround the scene. Their calibration is obtained offline through off-the-shelf libraries such as OpenCV. Each camera is handled by a dedicated PC. Each acquired image is locally analyzed to extract regions of interest (the foreground) which are then vectorized, i.e. their delimiting polygonal contours are computed.

3D modeling A geometric model is then computed from the silhouettes using an efficient method to compute the *visual hull* [13]. Obtained visual hull polyhedrons are sufficient for numerous VR applications including collision detection or virtual shadow computation for instance. To reach a real time execution, their computation is distributed among different processors.

Interactions and Visualization The 3D mesh is asynchronously sent to the interaction engines and to the visualization PCs. Multi-projector rendering is handled by a mixed replicated/sort-first approach.

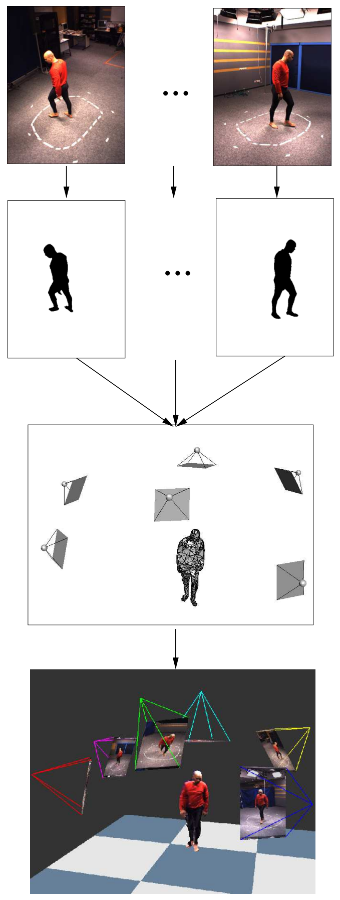


Figure 1. From multi-camera videos to dynamic textured 3D models

3 Acquisition

Acquisition takes place on a dedicated set of PCs, each connected to a single camera. These PCs perform all necessary preliminary image processing steps: color image acquisition, background subtraction and silhouette polygonalization (see figure 2). All cameras are standard firewire cameras, capturing images at 30 fps with a resolution of 780x580 in YUV color space.

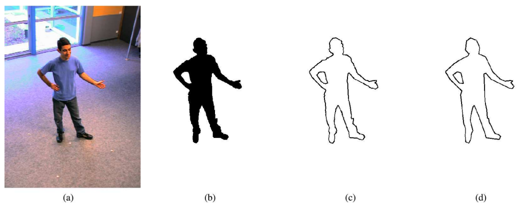


Figure 2. The different steps in the acquisition process: (a) the original image; (b) the binary image of the silhouette; (c) the exact silhouette polygon (250 vertices); (d) a simplified silhouette polygon (55 vertices).

3.1 Synchronization

Dealing with multiple input devices raises the problem of data synchronization. Indeed, our applications rely on the assumption that the input data chunks received from different sources are coherent, i.e. that they relate to the same scene event. We use an hardware synchronization where image acquisition is triggered by externally gen-locking the cameras, ensuring a delay between images below $100\mu s$.

3.2 Background Subtraction

Regions of interest in the images, i.e. the foreground or silhouette, are extracted using a background subtraction process. As most of the existing techniques [11, 7], we rely on a per pixel color model of the background. For our purpose, we use a combination of a Gaussian model for the chromatic information (UV) and an interval model for the intensity information (Y) with a variant of the method by Horprasert *et al.* [11] for shadow detection. A crucial remark here is that the quality of the produced 3D model highly depends on this process since the modeling approach is exact with respect to the silhouettes. Notice that a high quality background subtraction can easily be achieved by using a dedicated environment (blue screen). However, for prospective purposes, we do not limit ourself to such specific environments in our setup.

3.3 Silhouette Polygonalization

Since our modeling algorithm computes a surface and not a volume, it does not use image regions as defined by silhouettes, but their delimiting polygonal contours. We extract such silhouette contours and vectorize them using the method of Debled *et al.* [8]. Each contour is decomposed into an oriented polygon, which approximates the contour to a given approximation bound. With a single-pixel bound, obtained polygons are strictly equivalent to the silhouettes in the discrete sense (see figure 2-c). However in case of noisy silhouettes this leads to numerous small segments. A higher approximation bound results in significantly fewer segments (see figure 2-d). This enables to control the model complexity, and therefore the computation time, in an efficient way.

4 3D Modeling

The visual hull is a well studied geometric shape [13] which is obtained from a scene object's silhouettes observed in n views. It is the maximum shape consistent with all silhouettes. As such, it can be seen as the intersection of the images' *viewing cones*, the volumes that backproject from each view's silhouette (see figure 3).

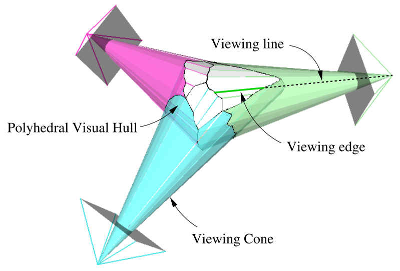


Figure 3. Visual hull of a sphere with 3 views.

We use a distributed surface-based method we have developed [9]. It recovers the exact polyhedral visual hull from the input silhouette polygons in three steps. First, a subset of the polyhedron edges – the viewing edges – is computed. Second, starting from this partial description of the polyhedron's mesh, all other edges and vertices are recovered by a recursive series of geometric deductions. Third, the shape's faces are recovered by traversing the obtained mesh. The following paragraphs briefly detail these steps, and their distribution over p CPUs.

4.1 Computing the Viewing Edges

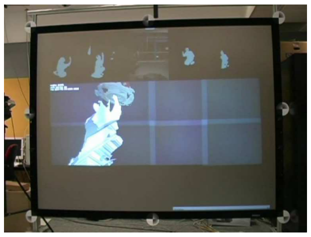
Viewing edges are intervals along viewing lines associated from silhouette contours' vertices. They are obtained by computing the set of intervals along such a viewing line that project inside all silhouettes. The distribution of this computation uses the fact that each viewing line's contributions can be computed independently. Viewing lines are partitioned into p identical cardinality sets and each batch is distributed to a different CPU. The final set is obtained by gathering partial results.

4.2 Computing the Visual Hull Mesh

The viewing edges give us an initial subset of the visual hull geometry. The missing chains of edges, are then recovered recursively starting from the viewing edges set. To allow concurrent task execution, the 3D space is partitionned into p slices. Slice width is adjusted by attributing a constant number of viewing edge vertices per slice for workload balancing. Each CPU computes the missing edges in its assigned slice. Partial meshes are then gathered and carefully merged across slice borders.

4.3 Computing the Faces

Faces of the polyhedron surface are extracted by walking through the complete oriented mesh while always taking left turns at each vertex, so as to identify each face's contours. Each CPU independently computes a subset of the face information, the complete mesh being previously broadcasted to each CPU.


(a) Carving

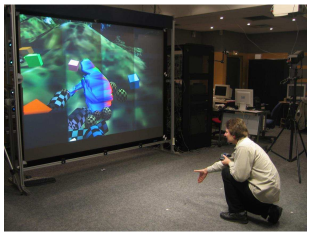

(b) Collision

Figure 4. Interaction experiments.

5 Interactions and Visualization

5.1 Real-Time interactions

We experimented two different interactions. The first one consists in a simple object carving (see figure 4(a)). The user can sculpt an object using any part of his body. This is done with octree-based boolean operations to update the object where it intersects with the user's model. Update operations include removal, addition of matter and change in sculpture color. The object can be rotated to simulate a potter's wheel.

The second interaction results from the integration of the user's model inside a rigid body simulation (see figure 4(b)). Several dynamic objects where added in the scene, and the system handles collisions with the user's body. This interaction requires all available information about the user's 3D surface, which is not available using classical tracking methods. Using our surface modeling approach, such fine level collision detection is something our system can achieve.

5.2 Multi-projector Visualization

To provide the user with a wide field of view while preserving image details, as necessary in semi-immersive and immersive applications, we have chosen to use a multi-projector display. The most scalable approach to implement this setup is to use one PC to drive each projector. To obtain a coherent image, each PC will have to synchronously render the same scene with a different view point, corresponding to the position of the related projector.

Several methods are available to implement parallel visualization, depending on the level of the primitives exchanged. We use a new framework [2], allowing to use a different scheme for each part of the scene. Large static objects, such as the landscape, use a replicated scheme so that they are sent locally on each PC. Other objects, such as the reconstructed mesh, are created on specific PCs and then sent to all visualization PCs, possibly culling invisible data (sort-first scheme).

The rendering of the 3D mesh itself is quite simple as it is already a polygonal surface. We can optionally compute averaged normal vectors at each vertex to produce a smoothly shaded rendering. It is relatively small (approximately 10000 triangles) so it can be broadcasted to all visualization PCs.

6 Implementation

6.1 The middleware library

To provide the I/O and computing power necessary to run our applications in real time, we use a PC cluster. However, coupling all pieces of code involved, distributing them on the PCs and insuring data transfers can be cumbersome. To get a high performance and modular application, we use a tool we developed [1], FlowVR, to manage distributed interactive applications. It relies on an data-flow model. Computation and I/O tasks are encapsulated into modules. Each module endlessly iterates, consuming and producing data. Modules are not aware of the existence of other modules. A module only exchanges data with the FlowVR daemon that runs on the same host. The set of daemons running on a PC cluster are in charge of implementing the data exchange network that connects modules. Daemons use TCP connections for network communications or shared-memory for local communications. The FlowVR network defined between modules can implement simple module-to-module connections as well as complex message handling operations like synchronizations, data filtering operations, data sampling, broadcasts, etc. This fine control over data handling enables to take advantage of both the specificity of the application and the underlying cluster architecture to optimize the latency and refresh rates.

6.2 Data-flow Graph

We propose for our application the following distributed data-flow graph from acquisition to rendering (see figure 5).

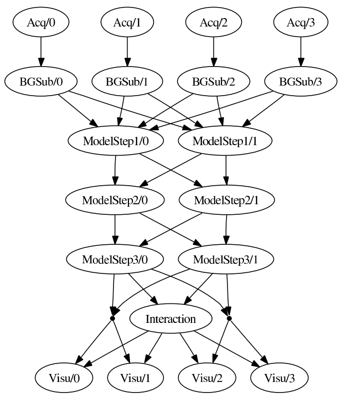


Figure 5. Data-flow graph from 4 cameras acquisitions to 4 video projectors rendering.

Each dedicated acquisition PC locally performs the data acquisition to obtain the silhouettes which are then broadcasted to the PCs in charge of the first modeling step, the viewing edge computation step. Follows the two other modeling steps, the global mesh recovery and the surface extraction. The resulting reconstructed surface is broadcasted to the PCs in charge of interaction computation and to the visualization hosts. These PCs also receive data from the interaction modules of the VR environment.

To obtain good performance and scalability it is necessary to setup specific coupling policies between the different parts of the application so they can run at different frequencies. The acquisition part typically runs at the frequency of the cameras while interactions run at more than $100Hz$. The visualization stage runs independently, allowing to change the viewpoint without waiting for the computation of the next 3D model. To implement these coupling policies we use two dataflow control policies: *FIFO* connections between modules running at the same frequency and *greedy sampling* connections (receivers always use last available data) between modules running asynchronously.

7 Results

We present the results obtained with our platform. It gathers 11 dual-Xeon 2.6 GHz PCs and 16 dual-Opteron PCs connected together by a gigabit Ethernet network. 6 FireWire Cameras are connected to the dual-Xeon machines. 16 projectors are connected to the dual-Opteron machines through NVIDIA 6800 Ultra graphics cards. The projectors display images on a flat screen of 2.7×2 meters. The acquisition space where the cameras are focused is located 1 meter away from the screen.

To evaluate the potential of 3D modeling for interaction purposes, we identified the following classical criteria as being relevant:

- Latency: it is the delay between a user's action and the perception of this action on the displayed 3D model. It is the most important criterion. A large latency can significantly impair the interaction experience. For most experiments on our system the overall latency, including all stages from video acquisition to visualization, was around $100ms$. This can be noticed by the user but is small enough to maintain a high level of interactivity. The quality of the background subtraction step as well as the simplification threshold applied to the resulting contours have a high impact on the latency as they determine the computational cost of the 3D modeling.
- Update frequency (modeling framerate): in our experiments, using 4 CPUs was enough to provide an update frequency of 30 Hz with 6 cameras when one user was in the interaction space.
- Quality (model's level of detail): in our experiments, the user was able to use its hands to carve virtual ob-

jects, and, depending on the angle relative to the cameras, it was possible to distinguish his fingers.

- Robustness to acquisition noise: our modeling algorithm is exact with respect to provided input silhouettes however noisy. The resulting 3D model is always watertight (no holes) and manifold (no self intersections). These properties are very important as many interaction applications or visualization (shadows, ...) rely on them. Moreover the approximation of silhouette contours removes most of the background subtraction noise.

- Model Content (the type of information available, surfaces, and textures in our case). When texturing the 3D models with the images obtained from the cameras, this property enables to avoid artefacs (see figure 6). Notice that in the applications presented the model is not textured. Real-time texturing is a challenging issue as the amount of data to handle in a distributed context is important. This is an ongoing work with very promising preliminary results.

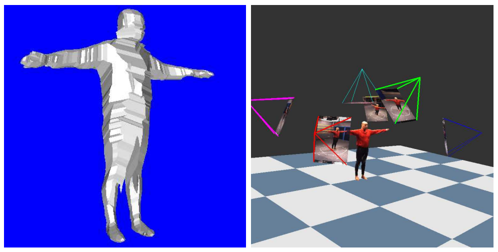


Figure 6. Details of a 3D model and its textured version (off-line).

8 Conclusion

We presented a marker-less 3D shape modeling approach which optimally exploits all the information provided by standard background subtraction techniques and produces watertight 3D models. The shape can easily be used for visual interactions, like rendering, shading, object occlusion, as well as mechanical interactions, like collision detection with other virtual objects. I/O devices and computing units are commodity components (FireWire cameras, PCs, gigabit Ethernet network, classroom projectors). They provide a scalable and efficient environment. The aggregation of multiple units and an adequate work-load distribution enable us to achieve real time performance.

Future works investigate two directions. One is to focus on data quality, in particular background subtraction and temporal consistency. The other is to focus on recovering semantic information about scene objects. The goal is to identify parts of the user's body for motion tracking, gesture recognition and more advanced interactions with the virtual world.

References

[1] FlowVR. `http://flowvr.sf.net`.

[2] J. Allard and B. Raffin. A shader-based parallel rendering framework. In *IEEE Visualization Conference*, Minneapolis, USA, October 2005.

[3] D. Arita and R.-I. Taniguchi. Rpv-ii: A stream-based real-time parallel vision system and its application to real-time volume reconstruction. In *Proceedings of ICVS, Vancouver (Canada)*, 2001.

[4] Eugene Borovikov and Larry Davis. A Distributed System for Real-time Volume Reconstruction. In *proceedings of CAMP-2000, IEEE*, 2000.

[5] J. Carranza, C. Theobalt, M. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. In *Proc. of ACM SIGGRAPH, San Diego*, pages 569–577, 2003.

[6] 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 CVPR, Madison*, 2003.

[7] 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 CVPR, Hilton Head Island*, volume 2, pages 714 – 720, June 2000.

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

[9] J-S Franco, C. Ménier, E. Boyer, and B. Raffin. A distributed approach for real time 3d modeling. In *Proceedings of the IEEE Workshop on Real Time 3D Sensors and Their Use*, Washington, USA, July 2004.

[10] J.-M. Hazenfratz, M. Lapierre, J.-D. Gascuel, and E. Boyer. Real Time Capture, Reconstruction and Insertion into Virtual World of Human Actors. In *Vision, Video and Graphics Conference*, 2003.

[11] T. Horprasert, D. Harwood, and L.S. Davis. A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection . In *IEEE ICCV'99 frame-rate workshop*, 1999.

[12] 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*, pages 1422–1428, 2000.

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

[14] M. Li, M. Magnor, and H.-P. Seidel. Improved hardware-accelerated visual hull rendering. In *Vision, Modeling and Visualization Workshop, Munich*, 2003.

[15] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image Based Visual Hulls. In *Proceedings of ACM SIGGRAPH*, pages 369–374, 2000.

[16] P.J. Narayanan, P.W. Rander, and T. Kanade. Constructing Virtual Wolrds Using Dense Stereo. In *Proceedings of ICCV, Bombay, (India)*, pages 3–10, 1998.

[17] W. Stephan, L. Edouard, and G. Markus. 3D Video Fragments: Dynamic Point Samples for Real-time Free-Viewpoint Video. *Computers & Graphics*, 28(1):3–14, 2004.

[18] X. Wu and T. Matsuyama. Real-Time Active 3D Shape Reconstruction for 3D Video. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, Rome (Italy)*, pages 186–191, September 2003.

6.9 Articulated Motion Capture from 3D points and Normals

Matti Niskanen, Edmond Boyer and Radu Horaud

Proceedings of the 16th British Machine Vision Conference

Oxford (UK), September 2005

Articulated motion capture from 3-D points and normals

Matti Niskanen[*]
Machine Vision Group
Infotech Oulu
University of Oulu
PO Box 4500, Finland

Edmond Boyer and Radu Horaud
Perception Group
INRIA Rhône-Alpes
655, avenue de l'Europe
38330 Montbonnot, France

Abstract

In this paper we address the problem of tracking the motion of articulated objects from their 2-D silhouettes gathered with several cameras. The vast majority of existing approaches relies on a single camera or on stereo. We describe a new method which requires at least two cameras. The method relies on (i) building 3-D observations (points and normals) from image silhouettes and on (ii) fitting an articulated object model to these observations by minimizing their discrepancies. The objective function sums up these discrepancies while it takes into account both the *scaled algebraic distance* from data points to the model surface and the *offset in orientation* between observed normals and model normals. The combination of a feed-forward reconstruction technique with a robust model-tracking method results in a reliable and efficient method for articulated motion capture.

1 Introduction

In this paper we address the problem of estimating the motion parameters of articulated objects, such as humans, from 3-D points and normals. These entities are inferred from 2-D silhouettes gathered with several synchronized cameras, Figure 1. The problem of tracking articulated shapes has been thoroughly studied in the recent past and a number of interesting methods and software packages are available. The vast majority of existing approaches and solutions relies on a single camera (a video sequence), on stereo (both binocular and trinocular), or on a large number of cameras. The first class of methods (a single video) attempts to recover the motion parameters directly from images and requires sophisticated probabilistic modelling. The second class of methods relies on depth data which, in turn, require search methods in order to solve for the stereo correspondence problem. The third class of methods relies on space-carving and level-set methods which are still under development. The latter has proved their usefulness for 3-D shape modelling but not for recovering motion parameters.

Here we describe a method which needs 2 to 6 cameras evenly distributed around the scene, i.e., they do not need to be arranged such that stereo correspondence is optimized. The method consists in fitting the pose of an articulated object model to 3-D observations gathered at some time instant, provided that the pose at the previous time instant has already been estimated. The object model is described by an *articulated implicit surface* that embeds a kinematic structure (such as a human body, a hand, an animal, etc.) and a set of volumetric primitives (ellipsoids).

[*]M. Niskanen is funded by Infotech Oulu. Financial support from Seppo Säynäjäkangas foundation is kindly aknowledged.

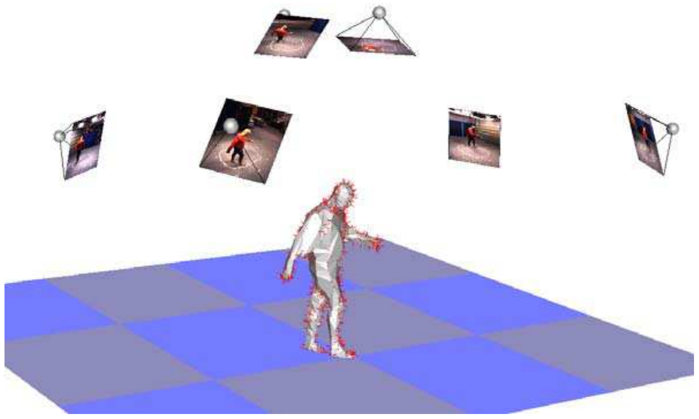


Figure 1: The cameras overlook a scene and a reconstruction method estimates 3-D points (connected to form a mesh for the purpose of the display) as well as 3-D vectors (shown as a needle field) normal to a smooth surface.

The implicit surface is defined as a distance function over these primitives and therefore this surface is simply a level set over a blending of these ellipsoids.

The 3-D observations are computed from image silhouettes gathered with the cameras. These 3-D data consist in surface patches, i.e., a 3-D point and a 3-vector. In order to fit these observations to the model we define a surface-patch-to-implicit-surface distance. The objective function to be minimized over the motion parameters is a sum of squares of the distances just mentioned.

Previous work. Since we adopt an "image understanding" point of view, we immediately rule out systems based on magnetic or optical markers, special-purpose clothes, and so forth. For a general review of human motion capture methods see [14]. Methods based on a single image sequence require a probabilistic framework [1], [8], and many others. An intrinsic difficulty, however, with methods based on 2-D data is the ambiguity of associating a multiple degree-of-freedom 3-D model with image contours, texture, and optical flow [4], [7]. Other researchers combine several cameras and make use of 2-D silhouettes whose image deformation is related, among others, to 3-D motion parameters. In [9], 2-D image data apply forces to a projected model and the parameters of the latter are adjusted such that the force field is minimized.

Methods using 3-D data are the most relevant with respect to our own approach. In general 3-D data are produced using stereo [5], [15], [6]. An articulated model based on cylindrical parts and an ICP algorithm is used in [5]. Both [15] and [6] use implicit surfaces defined over a set of spheroids, and these two methods are the most closely related to our own approach. In [15] an algebraic distance is minimized in order to fit the implicit surface to the depth data, and silhouette observations are used to constrain this surface to be tangent to rays originating at the optical center of the camera and passing through silhouette points. In [6] the stereo data are fitted to the model using an EM algorithm. Moreover, 3-D data that are consistent with the model are incrementally added to the latter such that both point-to-point and point-to-surface distance errors contribute to the fitting.

Original contributions. This paper has the following original contributions: First, 3-D observations (both points and normals) are computed from 2-D silhouettes based on multiple-camera geometric constraints and on the hypothesis that the observed 3-D surfaces are locally smooth; The method may well be viewed as an improvement over convex hull computation. There is no need to arrange the cameras such that stereo matching performs in an optimal manner. Second, the objective function, measuring the discrepancy between model and data, takes into account both point-to-surface and data-normal-to-model-normal discrepancies. We derive an analytic expression for these discrepancies which allows the straightforward implementation of non-linear minimization techniques. Third, the method avoids image projections of complex models. Fourth, data-to-model fitting is achieved in a single 3-D metric space instead of multiple, possibly inconsistent, 2-D projective spaces.

Organization. The remainder of this paper is organized as follows. Section 2 describes how 3-D data are obtained from image silhouettes. Section 3 describes the articulated model which is based on zero-reference kinematic chains, on ellipsoids, and on an articulated implicit surface defined over these chain and volumetric primitives. Section 4 describes the fitting between the data and the model based on both points and surface normals. Section 5 describes results obtained with both simulated and real data. Finally Section 6 draws some conclusions and suggests directions for future work.

2 Surface patches from image silhouettes

In this section we describe how 3-D points and surface normals are inferred from multiple image silhouettes. The 3-D shape data that we estimate consist in the positions of points and normals associated with the 3-D surface that produced the silhouettes. Such shape information is closely related to the visual hull of an object and it shares with the latter its robustness. Nevertheless, it is richer than the visual hull alone since it includes not only the surface tangent planes but also the surface positions which are not given by the visual hull. To estimate these positions, we use the fact that our surface models, ellipsoids, are C^2 surfaces. The method is valid, more generally, for locally smooth surfaces of order 2.

Viewing edges. We assume that a set of silhouettes – that segment the input images into foreground and background – are provided. These silhouettes may be combined to give rise to a *visual hull* which is the maximal 3-D shape consistent with them. The visual hull does contain the body surface and may intuitively be seen as the intersection of the *viewing cones* associated with the silhouettes. *Viewing edges*, or bounding edges [10], are the intervals along the viewing lines, as shown on Figure 2. They correspond to viewing-line contributions to the visual-hull surface and therefore they are associated to image points lying onto the silhouette boundary curves. Computing such a set of viewing edges is fast, simple, well-defined, and has already been used in various reconstruction applications, [12] and [3].

A silhouette is described by a discrete set of 2-D points. Viewing edges along a viewing line may be defined by combining silhouettes from two images and the associated epipolar constraint, as depicted in Figure 2. This can be easily extended to an arbitrary number of images and silhouettes. Whenever an additional silhouette from a new image is available, the viewing edges are updated to be consistent with contributions from the additional silhouette points. As the number of silhouettes increases, the length of the viewing edges narrows down.

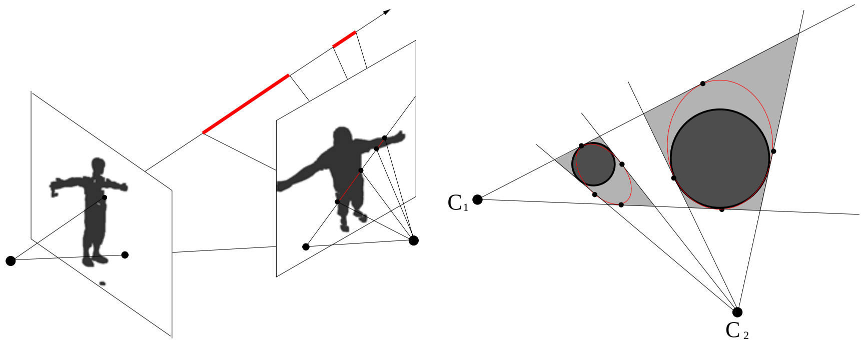


Figure 2: Left: two viewing edges along a viewing line computed solely from multiple-camera geometry. Right: two spheres (dark) that may be two distinct body parts, and the reconstructed surfaces (thin lines) with only two cameras. The *visual hull* is depicted by the shaded regions within the viewing lines originating in C_1 and C_2.

3-D points and surface normals. We explain now how to estimate the position and orientation of a *surface patch* that is supposed to lie onto the object's surface such that the latter is tangent to a viewing edge.

A viewing line associated with a silhouette from image j is tangent to the object's surface. If we assume that there is a unique viewing edge along a viewing line, then it means that this edge contains a surface point Y. Its orientation, a vector N, is defined by the cross-product between the viewing line and the tangent to the image silhouette. Notice that these computations can be carried out from image information only, provided that the calibration parameters of the camera are known.

The estimation of the position of point Y within a viewing edge requires some additional insights. Let Y belong to the viewing edge passing through the center of projection C_j of image j. This viewing edge is bounded by viewing lines associated with images i and k as well as their centers of projection C_i and C_k, as explained in the previous section. Since these viewing lines are tangent to the surface, we are also given these additional tangent directions – viewing lines originating in C_i and C_k – in the neighborhood of Y: The viewing lines from the silhouettes associated with images i and k which intersect the viewing line of Y . Under the assumption that the surface is locally of order 2, one can estimate the position of Y along a curve that lies onto the surface and which is constrained by three tangents. For farther details see [2].

The above reasoning applies to the case of a unique viewing edge along a viewing line. This is the case with most silhouette vertices if the cameras are evenly and sparsely distributed around the scene. However, this will not always be true, as shown on Figure 2. Whenever several viewing edges appear along the viewing line, the same approach is applied to each interval, one after one, thus producing as many 3D points and normals as the number of viewing edges. Note that not all the 3-D points thus determined actually belong to viewing edges tangent to the object's surface. Nevertheless, they all need to be computed in order to ensure that the local second order approximation of the surface is consistent with the visual hull. Moreover, as shown on Figure 2, the points thus obtained correspond to a better approximation of the object's surface than the visual hull itself. This is particularly important when the task is to fit a curved model to the observations. Results obtained with this method are shown on Figure 3.

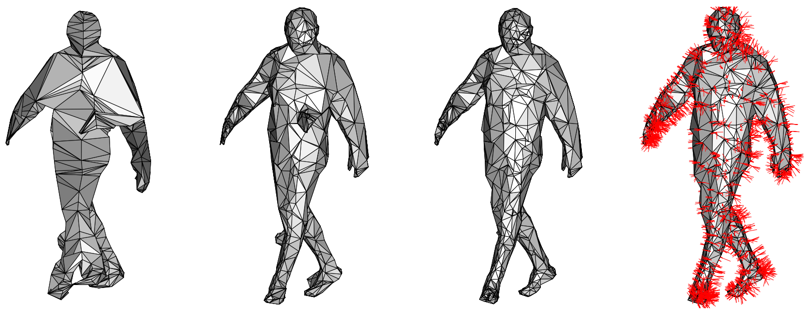


Figure 3: 3-D points (displayed as the vertices of a mesh) reconstructed with 2, 4, 5, and 6 cameras. The reconstructed normals are shown with the rightmost figure.

3 Modelling articulated objects

In order to model articulated objects such as human bodies, we use ellipsoids as basic volumetric shapes. These ellipsoids are joined and blended together to form an articulated implicit surface. In detail, an ellipsoid is a quadric described by a 4×4 homogeneous symmetric matrix $\mathbf{Q}$. This matrix is diagonal when the axes of the coordinate frame are aligned with the axes of inertia of the shape: $\mathbf{Q} = \text{Diag}\,(1/a^2\; 1/b^2\; 1/c^2\; -1)$. The implicit equation of its surface writes $X^\top \mathbf{Q} X = 0$ where X describes the homogeneous coordinates of a 3-D point lying on this surface. The *signed algebraic distance* from a data point Y to this surface is $q(Y) = Y^\top \mathbf{Q} Y$. The value of q varies from -1 at the origin, to 0 on its surface, and then to $+\infty$ outside the ellipsoid as the point is farther away from the surface. It is convenient to use the exponential of the algebraic distance as a measurement error. The scalar parameter σ bounds the *distance of influence* of an ellipsoid, i.e.:

$$r(Y) = \exp\left(-\frac{q^2(Y)}{\sigma^2}\right) \tag{1}$$

When an ellipsoid undergoes a rigid motion, its matrix becomes $\mathbf{Q}_T = \mathbf{T}^{-\top}\mathbf{Q}\mathbf{T}^{-1}$ where $\mathbf{T}$ denotes a 4×4 homogeneous matrix associated with an Euclidean transformation. $\mathbf{T}$ describes a *free motion*, a *kinematic chain*, or a combination of both. In our case the articulated object has rotational joints with either one or three degrees of freedom. Such a mechanism may be described by a kinematic chain of the form: $\mathbf{T}_1 \ldots \mathbf{T}_k \ldots \mathbf{T}_n$ where each individual transformation is a one-parameter Lie group that can be decomposed into a fixed transformation followed by a rotation around an axis aligned with the mechanical axis (or with a virtual axis), and followed by the inverse of the fixed transformation, $\mathbf{T}_k = \mathbf{L}_k \mathbf{J}(\theta_k) \mathbf{L}_k^{-1}$. Matrix $\mathbf{T}_k$ describes the position and orientation of joint axis k with respect to a reference frame, and:

$$\mathbf{J}(\theta_k) = \begin{bmatrix} \cos\theta_k & -\sin\theta_k & 0 & 0 \\ \sin\theta_k & \cos\theta_k & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The fixed part of this transformation, $\mathbf{L}_k$, depends on the particular length of the kth joint and on the position and orientation of this joint with respect to a fixed reference frame. Within this paper we do not address the problem of estimating the exact size and shape of the object's joints and therefore this transformation will be provided.

We also consider the free motion of the object, a matrix $\mathbf{D}$. In the case of a human body in motion we attach the body frame to the torso and we make the simplification that the free motion of the torso is a 3-D translation. Therefore, matrix $\mathbf{D}$ can be parameterized by three translations along three orthogonal directions, $\mathbf{D}_1\mathbf{D}_2\mathbf{D}_3$. Hence, the motion has n rotational joints and 3 free translations and is represented by $\Theta = (\theta_1 \dots \theta_n\ d_1\ d_2\ d_3)$; The motion transformation writes:

$$\mathbf{T}(\Theta - \Theta^0) = \mathbf{T}_1(\theta_1 - \theta_1^0)\dots\mathbf{T}_n(\theta_n - \theta_n^0)\mathbf{D}_1(d_1 - d_1^0)\mathbf{D}_2(d_2 - d_2^0)\mathbf{D}_3(d_3 - d_3^0) \tag{3}$$

This is known as the zero-reference representation of a kinematic chain because it describes the motion of each element of the object with respect to a fixed reference pose Θ^0 that can arbitrarily be chosen [13]. In the case of tracking, we seek the pose of the object at a time instant *t provided that the pose at the previous time instant $t-1$ has been already determined*, and therefore we can choose the pose of the object (and hence the pose of each one of its elements) associated with the previous time instant as the zero-reference pose: $\mathbf{T} = \mathbf{T}(\Theta^t - \Theta^{t-1})$. The matrix of an ellipsoid at time t can now be expressed as a function of the motion parameters, i.e., $\mathbf{Q}(\Theta^t) = \mathbf{T}^{-\top}\mathbf{Q}(\Theta^{t-1})\mathbf{T}^{-1}$.

We consider a complete object model. In particular a human body model with 22 rotational degrees of freedom is a relatively complete model that allows to capture the most general human actions. Therefore, there is a total of $22+3$ degrees of freedom, i.e., Θ is of dimension 25. Moreover, body parts are described by ellipsoids denoted by $\mathbf{Q}_1$, $\mathbf{Q}_2$, and so forth. Obviously, there is a kinematic chain for each body part and the number of degrees of freedom are different for each one of these chains. There is a quadratic form or a signed algebraic distance $q_i(X)$ associated with an ellipsoid $\mathbf{Q}_i$ as well as an exponential algebraic distance r_i, i.e., eq. (1); For an object in motion we have $q_i(X,\Theta)$ and $r_i(X,\Theta)$.

An articulated implicit surface can now be defined at each time instant as a level-set of a blending of these ellipsoids [15]:

$$f(X,\Theta) = \sum_{i=1}^{22} r_i(X,\Theta) = 1 \tag{4}$$

4 Fitting and tracking

It is now possible to formulate the problem of tracking an articulated shape as the problem of fitting the model to the data [6]. At each time instant the following minimization problem has to be solved:

$$\min_{\Theta} F(\Theta) = \left(\sum_{j=1}^{m} \beta_j (f(Y_j,\Theta) - 1)^2 \right) \tag{5}$$

where the weight β_j describes the probability of a data point Y_j to be consistent with the model, $\beta_j = \exp(-(f(Y_j,\Theta)-1)^2/\sigma^2)$. A large value for σ allows virtually all the data points to contribute to the fit, including data points that are far away from the model. A smaller value for σ allows to limit the influence of a datum to nearby quadrics. Within an Expectation-Maximization formulation such as in [6] an iterative procedure decreases the value of σ as the fitting proceeds. This allows, in principle, to escape from local minima when there is a large discrepancy between the data and the model pose. It also allows to disregard outliers at the final iterative steps of the algorithm. Surface orientation information is not taken explicitly into account. Ellipsoids whose local surface normals are very different will equally contribute when associated with a datum. We will modify the error function of eq. (5) in order to explicitly take into account *surface normals*.

The scaled algebraic distance. One important merit of any visual tracking method is its speed. Eventually tracking should be implemented in real time, i.e., compatible with the frame rates delivered by the cameras. Therefore, there is a compromise to be made between complexity and efficiency. The computation of the distance between an observation and the model resides in the inner loop of the tracker, and therefore it must be efficiently computed. The algebraic distance is fast to compute but has drawbacks. The Euclidean and pseudo-Euclidean distances are more expensive [6].

Let $\mathbf{Q}$ be an ellipsoid with parameters a, b, and c. Notice that matrices $\mathbf{Q}$ and $\lambda\mathbf{Q}$, with $\lambda \neq 0$ describe the same quadric. However the algebraic distances to these ellipsoids are different. Let $r^2 = a^2 + b^2 + c^2$. The scaled algebraic distance from a point Y to the ellipsoid is defined by $q^r(Y) = r^2 Y^\top \mathbf{Q} Y$. When the ellipsoid is close to a sphere and when the observation is close to its surface, the scaled algebraic distance is a good approximation of the Euclidean distance. However, with substantially elongated ellipsoids, the scaled algebraic distance does not introduce any improvements. Such an effect is known as high curvature bias. The practical solution that may be easily adopted consists in replacing elongated ellipsoids by an equivalent number of spheres.

Using surface orientation constraints. So far we used data points and we did not take into consideration the normals available with the 3-D observations. Let $N = (n_1\ n_2\ n_3\ 0)^\top$ be the vector normal to the surface patch and let $[N]_3$ denote the 3-vector formed with n_1, n_2, n_3. We also have $N^\top N = 1$.

It is well known that the 4-vector $P = \mathbf{Q}X$ defines the equation of a plane P tangent to the quadric at point X lying on its surface [11]. Therefore the 3-vector $[P]_3$ designates the normal vector to that plane. When a surface patch is consistent with the model, vectors $[P]_3$ and $[N]_3$ are aligned, therefore their cross-product is null and their dot-product is equal to either $+1$ or -1. A measurement of the discrepancy between a surface patch orientation and the nearby model orientation must use the followings:

$$d(Y,N,\mathbf{Q}) = [N]_3 \times [\mathbf{Q}Y]_3 \quad , \quad \alpha(Y,N) = \frac{1}{2}\left(1 - \frac{N^\top \mathbf{Q} Y}{\|\mathbf{Q}Y\|^2}\right)$$

The first one of these measurements, d, is equal to zero for a perfect match but is defined up to a 180^0 ambiguity. The second measurement, α varies between 0 (for vectors with opposite orientation) and 1; Therefore it may act as a normalized measure of a plausibility.

As in the case of point data, we define the exponential distance from an observation (a 3-D point Y and a normal N) to the 22 ellipsoids forming the model:

$$g(N,Y,\Theta) = \sum_{i=1}^{22}\left(\alpha_i \exp\left(\frac{-d(Y,N,\mathbf{Q}_i(\Theta))}{\mu}\right)\right) \tag{6}$$

Hence, one obtains an optimal solution by fitting all the 3-D observations to the model:

$$\min_{\Theta} G(\Theta) = \left(\sum_{j=1}^{m} (g(N_j, Y_j, \Theta) - 1)^2 \right) \qquad (7)$$

Tracking articulated objects. In order to track articulated objects we minimize a linear combination of the error functions $F(\Theta)$ and $G(\Theta)$; The first one of these functions, eq. (5), fits the locations of the observations with the model while the second one, eq. (7) fits the normals of the observations with the same model:

$$\min_{\Theta} (\omega_1 F(\Theta) + \omega_2 G(\Theta)) \qquad (8)$$

The tracking does not need segmentation of the data. Observations at time t are handled totally independently of observations at time $t-1$. The solution previously found, Θ^{t-1} is used in conjunction with a Kalman filter, and with a constant angular velocity hypothesis, in order to initialize the tracker at time t. Joint limits were set and added as penalty terms to the objective function in order to prevent unnatural human postures.

Another issue is the choice of ω_1 and ω_2 in eq. (8). These weights balance the contribution of position and orientation. There are methods allowing to initialize these weights and to modify them during the minimization process. However, as explained in the next section, we found that there are many advantages in using both position and orientation constraints. Therefore we chose $\omega_1 = \omega_2 = 1$.

5 Experiments

We validated the method with both simulated and real data. The former was obtained using a human animation software package. The latter was obtained with 6 calibrated cameras. Sequences of image silhouettes were generated with the animation software. Then the method described above was applied to these data. The simulated data allowed us to (i) assess the quality of the tracker with respect to a ground truth, (ii) analyse the behavior in the presence of Gaussian noise added to the data, (iii) quantify the merit of using surface normals, and (iv) determine the optimal number of observations needed to reliably estimate an object pose. Figure 4 illustrates some of the results out of a large number of experiments. From performing all these experiments one may conclude that tracking is notoriously improved when surface patches are used rather than just points. The surface-patch based objective function, i.e., eq. (8) converges faster, allows for less 3-D observations, and is more tolerant to errors in position. Figure 5 shows the results of applying the method to a 4 second sequence (120 frames) and with six cameras.

6 Discussion and conclusions

In this paper we described a method for tracking the motion of articulated objects. At each time instant, the images are segmented into foreground and background thus providing a set of 2-D silhouettes. These silhouettes are combined together with multiple-camera geometric constraints and with a simple assumption about the surface of the object in order to estimate 3-D surface patches: points and normals. The model itself is an *articulated implicit surface* combining a zero-reference kinematic chain with a set of ellipsoids. The model is fitted to the 3-D observation by minimization of an objective function that takes into account both the location

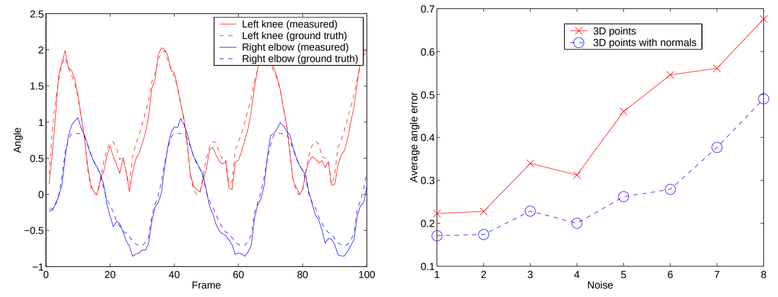


Figure 4: Left: comparison between the true angles and the estimated joint angles: left knee (top) and right elbow (bottom). Right: sensitivity to noise for points only (top) and for points and normals (bottom). The method always performs better when normals are taken into account. The two curves show the average angular error as a function of noise.

of these observations and their 3-D orientations. The resulting tracker is very efficient, it can deal with noisy data and with outliers, and it does not require data-to-object-part assignments.

Interesting enough, augmenting the number of cameras increases the robustness of the method without affecting its efficiency, since an increased number of cameras provides more precisely located surface patches. In practice we think that the optimal number of cameras is between 4 and 6.

Certainly, there are methods able to recover articulated motion with a single camera. These methods need sophisticated probabilistic methods to work well. They require a learning phase. We believe that our method is a potential candidate for providing data needed by learning methods.

In the future we plan to build a complete bio-mechanical model of humans with 80 degrees of freedom. We also plan to relax some of the constraints currently limiting our method, such as the requirement to have relatively accurate closed 2-D silhouettes. Finally, based on our fitting method, we plan to implement the bootstrapping of the tracker using a coarse-to-fine representation of the joint space and a hierarchical description of an articulated object.

References

[1] A. Agarwal and B. Triggs. Learning to track 3D human motion from silhouettes. In *International Conference on Machine Learning*, pages 9–16, Banff, July 2004.

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

[3] E. Boyer and J.-S. Franco. A hybrid approach for computing visual hulls of complex objects. In *Computer Vision and Pattern Recognition*, pages 695–701, June 2003. Madison, Wisconsin, USA.

[4] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara CA, pages 8–15, 1998.

[5] D. Demirdjian and T. Darrell. 3-D articulated pose tracking for untethered diectic reference. In *Proceedings of ICMI'02*, Pittsburgh, Penn., October 2002.

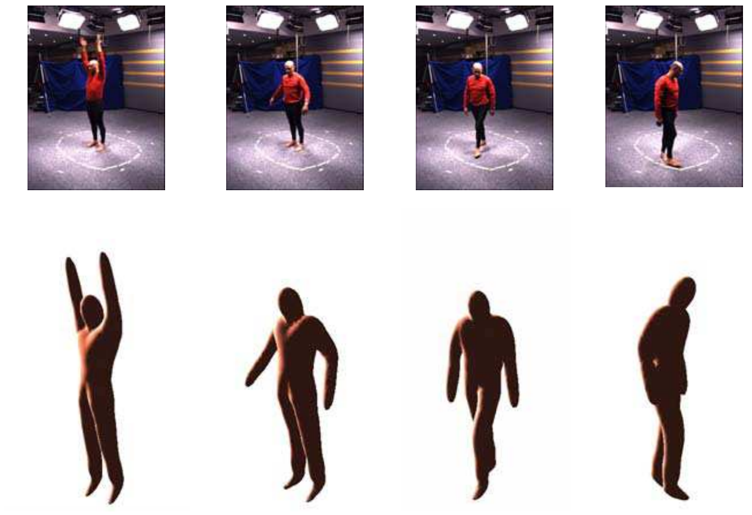


Figure 5: Top: Frames 17, 46, 59, and 81 for one of the six cameras. Bottom: The corresponding pose of the articulated implicit surface shown with the same camera parameters as above.

[6] G. Dewaele, F. Devernay, and R. Horaud. Hand motion from 3d point trajectories and a smooth surface model. In T. Pajdla and J. Matas, editors, *8th European Conference on Computer Vision*, volume I, *LNCS 3021*, pages 495–507. Springer, May 2004.

[7] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proceedings of the Eighth International Conference on Computer Vision*, volume II, pages 315–320, Vancouver, Canada, July 2001.

[8] N. R. Howe, M. E. Leventon, and W. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In *Advances in Neural Information Processing Systems*, Denver, volume 12, pages 820–826, 1999.

[9] I. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, 2000.

[10] 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 the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 714 – 720, June 2000.

[11] Q.-T. Luong and O. D. Faugeras. *The Geometry of Multiple Images*. MIT Press, Boston, 2001.

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

[13] J. M. McCarthy. *Introduction to Theoretical Kinematics*. MIT Press, Cambridge, 1990.

[14] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[15] R. Plänkers and P. Fua. Articulated Soft Objects for Multi-View Shape and Motion Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1182–1187, 2003.

6.10 3D Skeleton-Based Body Pose Recovery

Clément Ménier, Edmond Boyer and Bruno Raffin

Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission

Chapel Hill (USA), June 2006

3D Skeleton-Based Body Pose Recovery

Clement Menier Emond Boyer Bruno Raffin
GRAVIR–INRIA Rhône-Alpes
655, Avenue de l'Europe, 38334 Saint Ismier, France
E-mail: firstname.lastname@inrialpes.fr

Abstract

This paper presents an approach to recover body motions from multiple views using a 3D skeletal model. It takes, as input, foreground silhouette sequences from multiple viewpoints, and computes, for each frame, the skeleton pose which best fit the body pose. Skeletal models encode mostly motion information and allows therefore to separate motion estimation from shape estimation for which solutions exist; And focusing on motion parameters significantly reduces the dependancy on specific body shapes, yielding thus more flexible solutions for body motion capture. However, a problem generally faced with skeletal models is to find adequate measurements with which to fit the model. In this paper, we propose to use the medial axis of the body shape to this purpose. Such medial axis can be estimated from the visual hull, a shape approximation which is easily obtained from the silhouette information. Experiments show that this approach is robust to several perturbations in the model or in the input data, and also allows fast body motions or, equivalently, important motions between consecutive frames.

1. Introduction

An increasing number of virtual reality applications rely on marker-less interactions, for instance telepresence applications [16], or virtual object manipulation applications. This is, in most part, due to the fact that multi-view 3D modeling in real time becomes feasible, as demonstrated in recent works [8, 3]. However, models produced by such real-time methods are not necessarily rich enough to allow for complex interactions. In fact, information such as body part positions and velocities is often required by interaction applications. This *motion information* is related to, but different from, shape information for which efficient recovery solutions already exist. Our objective in this paper is therefore to focus on motion recovery, and in this way to provide a flexible and robust solution for body tracking from multiple views.

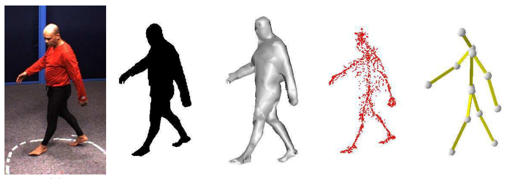


Figure 1. The tracking pipeline: (a) Color images ; (b) Silhouettes ; (c) Visual hulls ; (d) Medial axis points (d) ; (e) Skeleton pose.

Most marker-less motion tracking methods in computer vision fall into three categories. First, learning-based methods [1, 15] which rely on prior probabilities for human poses, and assume therefore limited motions. Second, model-free methods [9] which do not use any *a priori* knowledge, and recover articulated structures automatically. However, the articulated structure is likely to change in time, when encountering a new articulation for instance, hence making identification or tracking difficult. Third, model-based approaches which fit and track a known model using image information. In this paper, we aim at limiting as much as possible the required *a priori* knowledge, while keeping the robustness of the method reasonable for most interaction applications. Hence, our approach belongs to the third category.

Among model-based methods, a large class of approaches use an *a priori* surfacic or volumetric representation of the human body, which combines both shape and motion information. The corresponding models range from fine mesh models [6, 17, 4] to coarser models based on generalized cylinders [21, 12, 10], ellipsoids [8, 20] or other geometric primitives [11, 13, 14]. In order to avoid complex estimations of both shapes and motions as in [7], most approaches in this class assume known body dimensions. However, this strongly limits flexibility and becomes intractable with numerous interaction systems where unknown persons are supposed to interact. A more efficient solution is to find a model which reduces shape information. To this purpose, a skeletal model can be used. This model does not include any volumetric information. Hence, it has fewer dependencies on body dimensions. In addition, limbs lengths tend to follow biological natural laws, whereas human shapes vary a lot among population.

Recovering motion using skeletal models has not been widely investigated. Theobalt *et al.* [23] propose an approach where a skeletal structure is fitted with the help of hand/feet/head tracking and voxel-based visual hull computation. However, volumetric dimensions are still required for the arms' and legs' limbs. Luck *et al.* [19] also propose a method where skeletal arms are fitted to a voxel-based visual hull of the upper body. The method still requires knowledge of the body radius, and suffers from inadequate captured volumetric data. Brostow *et al.* [5] have proposed a model-free method based on the extraction of a skeletal structure from the user's shape. Our approach relies on this idea of using a skeletal structure but differs in the method to extract it and in the use of an articulated model.

In this paper, we propose to use a skeletal model and hence, to focus on body motion parameters in the model parameters. In this way, we allow for adaptability to body sizes without sacrifying robustness or time complexity with respect to the aforementioned approaches. A difficulty in this context is to find a relevant data space in which to fit the skeletal model. Our main contribution lies in the combination of the skeletal model with specific input data in the form of 3D medial axis points. These points are obtained by computing the medial axis of the visual hull shape associated with the body silhouettes in the images. Figure 1 depicts the different steps of the method. All these steps can be, in the short term, achieved in real time, which makes the approach a good candidate for real time interaction applications.

§ 2 describes our skeletal articulated model and § 3 the associated measured data. § 4 presents the fitting and tracking process. § 5 reports on results obtained for real sequences and discusses on real time performance issues before concluding in § 6.

2. Skeletal Articulated Model

In this section, we describe the *a priori* articulated model representing a body pose. A great variety of models have been proposed in the literature. They rely on a kinematic chain adjoined with a shape model of the person (ellipsoids, quadrics, generalysed cylinders, *etc.*). Those models are thus specific to a particular user. We propose instead to use a 1D articulated model, therefore not including any volumetric information on the user.

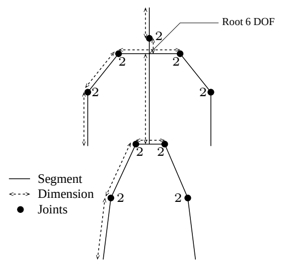


Figure 2. The skeletal articulated model.

This skeletal articulated model consists in a kinematic chain of segments. As interactive applications are usually only interested in the principle joints (elbows, shoulders, knees, legs and head), we limit our model to a set of 12 segments with those 9 joints (see figure 2). This leads to 24 degrees of freedom: 2 per joints and 6 for the root position and orientation. Note that other models, with higher fidelity to the human anatomy, could also be used if required by more demanding applications (e.g. graphics animations). For joints having 2 degrees of freedom, we chose a representation based on Euler angles. To avoid the classical discontinuity problems encountered with Eulerian parametrizations, we set the axis of rotation (where singularities occur) in the most unlikely direction (due to natural joint constraints for example). This proved to be sufficient in most of our experimentations. Other parametrizations, such as quaternions, would not necessarly give better results since they represent full 3D rotations (3 degrees of freedom).

3. Observed Skeleton Data

Another important element of the tracking process is the data which is considered as the measurement for the body pose, and to which the model is fitted. A great variety of data has been proposed in the literature for that purpose.

[14, 17, 6] use 2D cues such as silhouettes or contours. The body model is projected onto available image planes, and the fitting is achieved in the 2D image spaces. This has 2 major drawbacks: first, image features only affect the corresponding visible parts of the body model which must first be identified; second, skeletons are not invariant by projection, i.e. the 3D skeleton of a shape does not project onto the 2D skeletons of the projected shape, and thus fitting the projection of a 3D skeleton to 2D skeletal data, such as 2D medial axis, would not make sense.

Other aproaches have proposed to directly use 3D cues. Most of them consider 3D data resulting from multi-view modeling methods such as Shape-From-Silhouette [4, 19] or stereo [11]. Such shape information is particularly well adapted when fitting shape models such as ellipsoids [8]. However it is not adapted to our approach since skeletal and shape information are of different nature and fitting our model to shape data would necessarly lead to inconsistencies.

More recently, Brostow *et al.* [5] have proposed to use 3D skeletal information for motion analysis. They retrieve motion information directly from an extracted 1D skeletal structure. Their approach being model free, a great care is taken to obtain a very precise skeleton, leading to a very slow extraction (several minutes per frame). It is therefore not adapted for interactive systems, which is our main objective. We propose to use a less robust but faster skeleton extraction technic. The lack of precision in the skeleton extraction is compensated by the *a priori* knowledge (human articulated model).

In our approach, we assume that silhouettes, extracted from calibrated cameras with different viewpoints, are available. These silhouettes are obtained through standard background subtraction methods. From these silhouettes, we first compute their 3D equivalent, i.e., the visual hull [18]. To this purpose, we use an exact method [3] which computes a polyhedron in space. This shape exactly projects onto the silhouettes in the images and thus preserves all the silhouette information. It is then processed in order to extract its internal structure, namely a skeleton. This step, called skeletonization, has received considerable attention from the computational geometry community. Several definitions can be considered for skeletons but the most successful is certainly the medial axis [22]. The medial axis is defined as the locus of centers of closed balls that are maximal with respect to inclusion. In the case of a discrete surface, the process leading to a discrete approximation of the medial axis is sometimes called the Medial Axis Transform. An important drawback of the discrete medial axis comes from its sensitivity to noise (see figure 3(b)). However some works have tackled this issue and proposed algorithms that take into account input shape noise. Attali *et al.* [2] have proposed such an algorithm. The idea is first to compute a discrete medial axis and second to prune it in order to eliminate outliers. The algorithm proceeds then as follows:

1. Voronoi centers are computed from the mesh vertices. Note that we only consider centers lying inside the mesh (see figure 3(b)).
2. For each center C we retrieve its corresponding Delaunay tetrahedron (P_1, P_2, P_3, P_4) and compute:
   - its radius $\rho(C) = d(C, P_1)$
   - its bisector angle $\theta(C) = \max_{i \neq j}(\widehat{P_i C P_j})$.
3. Outliers are eliminated based on a minimal radius and bisector angle threshold.

This results in a set of 3D points $\{X_0 \cdots X_n\}$ that we call the *Skeleton Data* (see figures 3(c) and 3(d)). Chosing a radius and bisector angle threshold consists in finding a tradeoff between the skeleton quality and the number of resulting points. Indeed the higher the thresholds are, the better the skeleton is but the fewer points are selected (see figure 3(d)). In practice we set the radius threshold at 4 cm and the bisector angle threshold around $160°$ (see figure 3(c)). It should be noticed here that the 3D medial axis is not a curve, as in 2D, but a surface. In practice, this has little impact on our approach for 2 reasons. First, the width of this surface, in the human case, is usually less or at most comparable (in the case of the torso) to the measurement noise. Second, the skeletal structure lies at the middle of the medial axis surface, therefore minimizing distances to the extracted medial axis points. Note that other skeletonization methods may be used, such as Brostow's method, as our fitting method is not specific to the medial axis but to the expense of the interactivity of the system.

4. Model Tracking

We have defined, in the previous sections, our skeletal articulated model and the observed skeleton data. In this section, we first define the generative model which explains the observations in function of the articulated model. We then present how this generative model is used in a fitting process which computes the maximum a posteriori estimate (MAP). Finally we discuss the important issue of pose tracking over sequences.

4.1. The generative model

In order to retrieve the pose of the user at a given time t, we must define the relationship between the *a priori* articulated model and the observed data. A first solution would be to characterize the similarity between the skeleton dataset of points $\{X_0, \cdots, X_n\}$ and a skeletal model S based on the distance of each point to its closest articulated segment $s \in S$ as in the following joint probability:

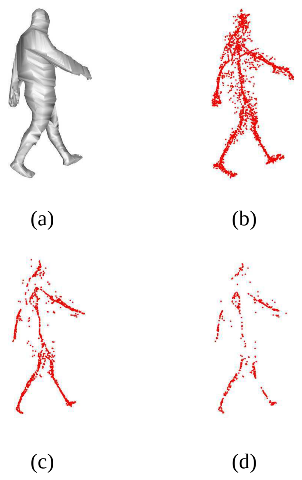


Figure 3. (a) Exact visual hull obtained. (b) Internal voronoi centers yielding a noisy skeleton. (c) Skeletonization after pruning with $r > 4$ cm and $\theta > 160°$: most outliers are removed. (d) Skeletonization after pruning with $r > 5$ cm and $\theta > 170°$.

$$P(\{X_i\}\ S) = P(S) \times \prod_{i=0}^{n} P(X_i|S)\ , \qquad (1)$$

where: $P(X_i|S) = \mathcal{N}(d(X_i, S), \sigma^2)$ and $d(X_i, S) = \min_{s \in S} d(X_i, s)$, with $d()$ representing the Euclidean distance.

However, maximizing the corresponding posterior distribution $P(S|\{X_i\})$ leads to difficulties. Indeed, the attachment of a point to a segment is subject to change during the fitting process, generating inconsistencies and gradient discontinuities. To solve this issue, we introduce hidden variables a_i, one for each point, representing the segment attached to point X_i. The joint probability of the observed data and the pose becomes then:

$$P(\{X_i\}\ \{a_i\}\ S) = P(S) \times \prod_{i=0}^{n} P(a_i|S) \times \prod_{i=0}^{n} P(X_i|a_i\ S),$$

where:

- $P(S)$ is the prior distribution of the pose. In our case we make the assumption of an uniform distribution. However it could account for joint constraints and/or knowledge on given poses (splits are less probable than standing positions for example).
- $P(a_i = j|S)$ represents the *a priori* on the attachment with the sole knowledge of the pose. We set it proportional to the length of the corresponding segment s_j. Note that with our model, the segment lengths are fixed. Hence, this prior distribution does not depend on the pose.
- $P(X_i|a_i = j\ S)$ represents the probability that point X_i belongs to the limb corresponding to the segment s_j. We model it as a standard gaussian $\mathcal{N}(d(X_i, s_j), \sigma_j^2)$. Note that with an ideal skeletonization algorihm, all σ_j should be identical (uniform noise). However in practice, skeletonization methods lead to higher noise on the torso than on the arms or the legs. The variances σ_j are therefore set to approximately 1 cm, except for the torso where it is set to approximately 3 cm.

Finding the best pose consists then in maximizing the following posterior:

$$\begin{array}{rcl} P(S|\{X_i\}) & \propto & \sum_{\{a_i\}} P(\{X_i\}\ \{a_i\}\ S), \\ & \propto & \prod_{i=0}^{n}\ \sum_{a_i} P(X_i a_i S), \\ & \propto & P(S) \prod_{i=0}^{n}\ \sum_{a_i} P(a_i|S) P(X_i|a_i S). \end{array}$$

Unlike the first solution (1), this posterior is well adapted for maximization as all its derivatives are continuous ($\mathcal{C}^\infty$ function). This posterior is also more robust as it marginalizes over all possible point to segment attachments instead of considering the single possible attachment from a point to its closest segment.

4.2. Fitting

In order to find the above MAP and as classical when dealing with hidden variables, we use an expectation maximization approach where:

- **The E step** consists in the computation of the expectation terms $E(a_i = j)$ for the current estimated pose $\overline{S}$:

$$\begin{array}{rcl} E(a_i = j) & = & P(a_i = j|X_0 \cdots X_n\ \overline{S}), \\ & = & P(a_i = j|X_i\ \overline{S})\ , \\ & = & \frac{P(a_i = j\ X_i\ \overline{S})}{\sum_{a_i} P(a_i\ X_i\ \overline{S})}\ ; \end{array}$$

- **The M step** consists in finding the pose S maximizing:

$$F(S) \quad = \quad \sum_{i=0}^{n} \sum_{a_i} E(a_i) \times logP(a_i\ X_i\ S).$$

Developping $P(a_i X_i S) = P(S)P(a_i|S)P(X_i|a_i\, S)$, we notice that the two first terms are constants. $P(S)$ is supposed uniform and the prior distribution on a_i does not depend on the pose. This leads to maximizing:

$$F(S) \propto \sum_{i=0}^{n} \sum_{a_i} E(a_i) \times log P(X_i|a_i\, S)$$

This is equivalent to minimizing its negated form:

$$\sum_{i=0}^{n} \sum_{a_i=j} E(a_i = j) \times \frac{d(X_i, s_j)^2}{2\sigma_j^2}$$

This formula defines a least squares problem. We use the well known Levenberg-Marquardt minimization algorithm as it is well adapted to this type of problem.

4.3. Tracking

The fitting process recovers a single pose at a given frame. To recover the motion of the user, we need to describe how we obtain the pose S_{t+1} at frame $t+1$ knowing the previous poses. This "propagation" problem consists in predicting a likely position S'_{t+1}. This prediction is used as an initial guess in the minimization process resulting in the final pose S_{t+1}. This prediction is commonly based on a dynamic model such as constant velocity or constant acceleration. Those models are efficient in modeling displacements of objects with relatively stable velocity. This condition generally implies a small ratio between the applied forces and the mass of the object. If this condition is valid for the root position and orientation of the body, it is clearly not valid for arms or legs. Their motions can be very erratic. In such cases tracking without dynamic model ($S'_{t+1} = S_t$) is a good solution as our experiments will demonstrate. A better solution would be to consider that the recovered velocity is noisy and incorporate a noise model in the propagation process with a particle filtering or belief propagation algorithm for example. In our experiments however particle filtering with up to 1000 particles did not improve results while significantly increasing the computational cost. We therefore seldom use it. Using non parametric belief propagation could lead to better results but again this would make the tracking process too slow for interactive systems.

5. Results

The body tracking method presented in the previous sections has been implemented and tested on various sequences of natural motions like walking in any direction. In this section, we present the corresponding results and discuss the robustness of our tracking method. We also discuss an important issue which is time performance through computations cost.

5.1. Data Acquisition

Image sequences were acquired using 6 firewire cameras shooting 780×580 images at 27 fps. These cameras are electronically triggered to ensure synchronization between images. Silhouettes are obtained through a standard background subtraction method. Results shown here are based on 2 sequences. The first one consists in a person walking in circle and lasts 15 seconds (around 400 frames), corresponding to 2 walking circles. The second one consists in a person performing a rapid kick in the air. It lasts 4 seconds with only 30 frames corresponding to the kick itself. Dimensions of the model were manually set, with an error of approximately 10%.

5.2. Tracking Results

Tracking results on the walking sequence are presented in figure 4. Validation is done by visually checking each frame. Only 6 frames out of 400 were found partially mistracked. Those 6 frames are organized in 2 groups of 3 consecutive frames, the 2 groups corresponding to the same situation in the sequence but at different times. In this situation, an elbow joint was found away from its real position (see figure 4-frame 290 for instance). This situation is due to visibility problems which result in skeleton data outliers between the torso and the arm that are wrongly attached to the arm, making the elbow moving toward the torso. Note that such a situation could probably be avoided by using temporal consistency through a dynamic model, again to the price of computational cost.

Tracking results on the kicking sequence are presented in figure 5. This sequence was used to evaluate the robustness of the approach to large motion between consecutive frames, or in other words to fast motions with respect to the acquisition frame rate. As shown by the results, the approach behaves well in such situation, even without prediction between consecutive frames, validating in that case the fact that no dynamic model is used.

5.3. Robustness

An important aspect for body tracking algorithms concerns their robustness to all types of noise. In our case, the main sources of errors are coming from noises in the input data as well as in the model parameters. Both are discussed in this section.

Noisy input data

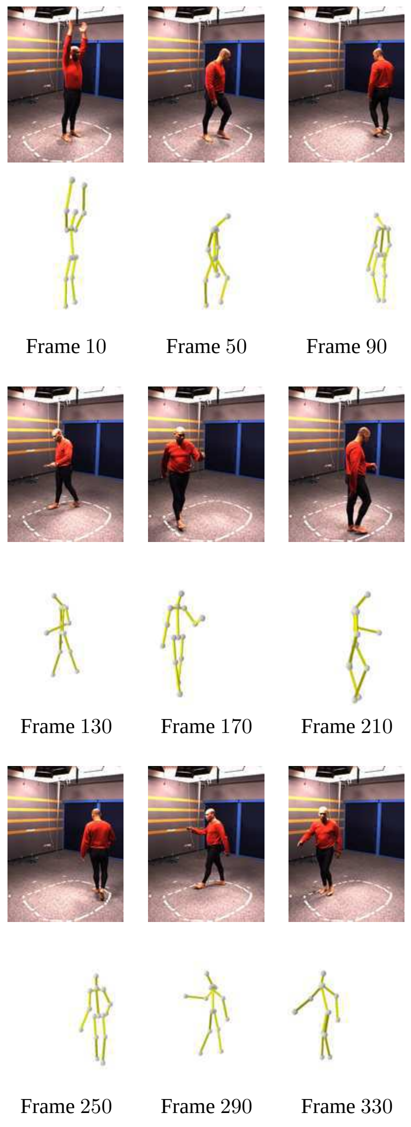


Figure 4. Skeleton poses at different times for the walking sequence.

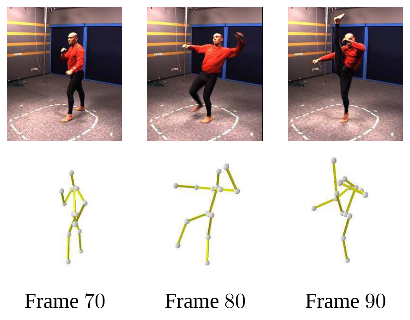


Figure 5. Recovered body poses for the kicking sequence.

Sequences are not taken in specific environments, such as blue rooms, resulting in noisy silhouettes as obtained by background subtraction (see figure 6). Our approach is robust to those errors in different ways. First, notice that since the visual hull algorithm used is exact with respect to silhouettes, it does not add any additional noise but filters instead silhouette errors which are inconsistent in different views (or false positives). Second, the medial axis is pruned which allows for some errors in the shape estimation.

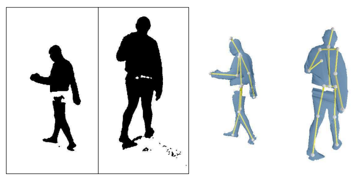


Figure 6. Left, examples of noisy silhouettes in the sequence. Right, result of the skeleton pose estimation with these silhouettes (2 different viewpoints).

Robustness to model errors

To test errors in the *a priori* model, noise was introduced in the dimensions of the model used for the walking sequence. The tracking performs correctly (only few partial mistracked frames) up to 20% of error. For higher noise, the number of mistracked frames increases: 30 frames out of 400 are mistracked in the walking sequence with 30% of error in the model. This robustness to model dimensions errors and the fact that the ratio between a limb size and the height of a person does almost not vary among the global population enables the model to be determined by only the height of the human body. This idea is currently being validated on a set of sequences acquired with users presenting different morphologies.

5.4. Real Time Issues

One of the main constraints imposed by interactive applications is real-time performances associated with low latencies. In this section, we discuss this issue for the two main steps of our method.

Skeleton Data Computation: As demonstrated in [3], the visual hull computation can be achieved in real time with a latency of 70 ms. The skeletonization cost lies essentially in the voronoi cells computation. This takes about 60 ms for 2000 surface points on an Opteron 2GHz. Distributing its computation allows this process to run at 30 frames per second but does not reduce its latency. Real time performance – less than 30 ms – is likely to be achieved in a year with the growth of computational power.

Tracking: Our tracking takes about one second per frame. Most of the time is spent computing distances from points to the model segments. This could be reduced by considering that only the 2 or 3 closest segments are relevant. This would reduce the computational cost by a factor of 5. Note also that this implementation is only an experimental prototype. Code optimization could significantly reduce computational cost. Moreover the *a posteriori* function can largely benefit from parallelization on multiple CPUs, as skeleton data input points can be treated independandly.

6. Conclusion

We have presented a 3D tracking algorithm that focus on motion parameters and relaxes dependencies on body shapes. It is based on a skeletal articulated model which is fitted to 3D skeleton data points. Those points lie on the medial axis of the visual hull, as obtained from silhouettes in multiple views. Experimental results on real sequences have been presented. They demonstrate the robustness of our approach to different aspects such as silhouette noise or dimension errors. This approach is relatively fast and should reach real time performances in the near future.

Several issues still remain to be addressed. First temporal consistency could be taken into account. One solution could be to integrate it directly in the generative model by changing $P(S)$ by $P(S_t|S_{t-1})$ corresponding to the probablistic dynamic model. Additionnaly, the uniform hypothesis on $P(S)$ could be changed to allow various joint constraints and to ensure that the skeletal model lies inside the visual hull. Second, the robustness of the tracking can be improved. In particular, the points to segments association could be more efficient if the visual hull containment constraint was taken into account. This would prevent attachment between torso points and arms for example. Also multiple cues such as color information (appearance model) or head/hand 3D tracking could be integrated in the process.

References

[1] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *Proceedings of CVPR'O4, Washington, (USA)*, pages II 882–888, June 2004.

[2] D. Attali and A. Montanvert. Modeling noise for a better simplifi cation of skeletons. In *Proceedings of ICIP, Lausanne (Switzerland)*, 1996.

[3] Authors.

[4] A. bottino and A. Laurentini. A silhouette based technique for the reconstruction of human movement. *Computer Vision and Image Understanding*, 83:79–95, 2001.

[5] G. J. Brostow, I. Essa, D. Steedly, and V. Kwatra. Novel skeletal representation for articulated creatures. In *Proceedings of ECCV'04, Prague*, pages Vol III: 66–78, 2004.

[6] J. Carranza, C. Theobalt, M Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *Proc. ACM Siggraph'03,* San Diego, USA, pages 569–577, July 2003.

[7] G. Cheung, S. Baker, and T. Kanade. Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture. In *Proceedings of CVPR'O3, Madison, (USA)*, 2003.

[8] 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 CVPR'00, Hilton Head Island, (USA)*, volume 2, pages 714 – 720, June 2000.

[9] C.-W. Chu, O. C. Jenkins, and M. J. Matarić. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of CVPR'03, Madison, (USA),* pages 475–482, 2003.

[10] I. Cohen, G. Medioni, and H. Gu. Inference of 3d human body posture from multiple cameras for vision-based user interface. In *5th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando*, 2001.

[11] Quentin Delamarre and Olivier D. Faugeras. 3d articulated models and multi-view tracking with silhouettes. In *Proceedings of ICCV'99, Kerkyra, (Greece)*, pages 716–721, 1999.

[12] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle fi ltering. In *Proceedings of CVPR'00, Hilton Head Island,* pages 2126–2133, 2000.

[13] T. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proc. of ICCV'01, Vancouver*, pages 315–320, 2001.

[14] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *Proceedings of CVPR'96, San Francisco)*, pages 73–80. IEEE Computer Society, 1996.

[15] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. In *Proceedings of ICCV'03, Nice, (France)*, pages 641–648, 2003.

[16] M. Gross, S. Wuermlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, K. Strehlke S. Lang, A. Vande Moere, and O. Staadt. Blue-c: A spatially immersive display and 3d video portal for telepresence. In *ACM SIGGRAPH 2003*, San Diego, 2003.

[17] I. Kakadiaris and D. Metaxas. Model-based estimation of 3d human motion. *IEEE Transactions on PAMI*, 22(12):1453–1459, december 2000.

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

[19] J. Luck, D.Small, and C. Q. Little. Real-time tracking of articulated human models using a 3d shape-from-silhouette method. In *Roboton Vision*, pages 19–26, 2001.

[20] Ivana Mikić;, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Human body model acquisition and tracking using voxel data. *IJCV*, 53(3):199–223, 2003.

[21] A. Senior. Real-time articulated human body tracking using silhouette information. In *proceedings IEEE Workshop on Visual Surveillance/PETS, Nice, France*, october 2003.

[22] J. Serra. *Image Analysis and Mathematical Morphology, Volume I*. Academic Press, 1982.

[23] C. Theobalt, M. Magnor, P. Schüler, and H.-P. Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. *International Journal of Image and Graphics*, 4(4):563–584, October 2004. Pacifi c Graphics 2002.

Chapitre 7

Bilan et perspectives

Les travaux présentés dans ce document traitent de la modélisation de scènes dynamiques à partir de plusieurs flux vidéos. Sur ce thème nous avons parcouru ces dernières années un chemin certain. Nous sommes en effet passé de la modélisation statique tel que cela était possible il y a quelques années à une architecture capable de gérer de façon automatique plusieurs flux vidéos, cela de manière flexible en s'appuyant sur la puissance de calcul modulable d'une grappe de PCs. Pour atteindre ce but, nous avons résolu de nombreux problèmes et proposé des solutions originales que ce soit sur la compréhension du problème de modélisation à partir de silhouettes, sur le calibrage ou sur la capture de mouvement. Ces contributions ont fait l'objet de publications dans les grandes conférences internationales du domaine de la vision par ordinateur. La réalisation technologique associée, concrétisée par la plate-forme GrImage, fait actuellement l'objet d'un transfert vers une société et nous avons bon espoir qu'elle trouve une voie commerciale, dans le domaine de la cinématographie 3D notamment, au cours des prochaines années.

Le chemin n'est pour autant pas à son terme et de nombreuses directions scientifiques restent à explorer que ce soit pour améliorer les modèles obtenus ou mettre à jour de nouveaux axes. Une première direction que nous n'avons pas explorée dans ce document concerne l'information photométrique ou encore l'apparence. Cette information permet en effet d'améliorer de manière significative les modèles obtenus à partir de silhouettes. Plusieurs approches ont déjà été proposées, notamment [Her 04, Sin 05, Fur 06], dans le cas de scènes statiques et lorsque de nombreux point de vue sont disponibles. Le cas de vidéos, issues d'un nombre éventuellement réduit de caméras, reste à considérer. Cela mène naturellement à une deuxième direction qui concerne la dimension temporelle des informations que nous traitons. Cette dimension n'est que peu considérée pour le moment, les modèles sont effet construit indépendemment à chaque instant. Or, les informations sur une scène, dynamique ou non, sont redondantes dans le temps et cette *cohérence temporelle* devrait être exploitée. Cela concerne l'information géométrique et photométrique. L'idée serait donc d'améliorer un modèle, en terme de forme et d'apparence, au fur et à mesure dans une séquence temporelle d'images. Un des défis à relever pour cela est de pouvoir mettre en correspondance les informations dans le temps afin de leur appliquer des règles de cohérence. Il s'agit, par exemple, de trouver pour un point sur une forme à l'instant t le correspondant sur la même forme à l'instant $t + 1$. Un autre défi en lien direct est celui de la représentation et de l'estimation du mouvement que ce soit le mouvement d'un objet rigide, d'un objet articulé ou d'un objet déformable. Ce problème reste fondamental en vision par ordinateur et encore mal résolu, même lorsque des connaissances *a priori* sur les objets observés sont disponibles.

Une autre direction novatrice concerne l'interprétation de scènes dynamiques au travers de vidéos. La capacité pour un système de reconnaître une situation et d'y réagir ouvre des perspectives nombreuses, notamment pour les applications de vidéo surveillance ou les applications interactives. Des vidéos multiples constituent une riche source d'informations qui peut fournir différents types de données sur les activités dans une scène observée, des primitives images de bas niveau jusqu'à des primitives tridimensionnelles de mouvement par exemple. Un des challenges qui se présentent alors est d'organiser l'ensemble des informations disponibles, issues de plusieurs vidéos et éventuellement d'autres capteurs, en des primitives cohérentes qui peuvent être classées en catégories de gestes, d'actions ou d'activités. Nous travaillons actuellement sur ce problème de représentation, en particulier dans le cas d'activités humaines, avec l'objectif de proposer des modèles tridimensionnels et donc indépendants du point de vue, qui permettent la constitution de librairies utiles à la reconnaissance. Un autre challenge réside dans les méthodes statistiques à mettre en oeuvre pour la reconnaissance, notamment en regard des dimensions des espaces à explorer. À ce titre, une direction intéressante consiste à réduire ces dimensions en identifiant le contexte dans lequel se déroulent les activités observées. Le contexte, des objets par exemple, permet en effet de limiter les activités ou actions observées à un sous-groupe. Nous travaillons actuellement sur ces aspects.

Enfin mentionnons une autre direction qui concerne l'aspect technologique des travaux présentés dans ce document. La mise en oeuvre de la plate-forme expérimentale GrImage a en effet était un vecteur de collaborations scientifiques riches et une source foisonnante de problèmes de recherche. Elle est à ce jour à un degré de maturité suffisant pour considérer une exploitation commerciale mais son évolution reste, à mon sens, un enjeu important pour les thèmes de recherche évoqués dans ce document. Cette évolution est orientée vers les environnements intelligents où la perception reposera sur des sources multiples d'informations, et non seulement des caméras, et où l'action pourra être non seulement visuelle, comme c'est le cas avec la plate-forme actuelle, mais plus effective. Les problèmes qui se posent alors sont bien sur multiples et doivent être considérés dans une perspective à plus long terme que les directions mentionnées précédemment.

Bibliographie

[Aga 04] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington, (USA)*, pages II 882–888, June 2004. (citation page 31)

[All 05] J. Allard, C. Ménier, E. Boyer, and B. Raffin. Running Large VR Applications on a PC Cluster : the FlowVR Experience. In *Immersive Projection Technology*, October 2005. (citation page 3)

[All 06] J. Allard, J.-S. Franco, C. Ménier, E. Boyer, and B. Raffin. The GrImage Platform : A Mixed Reality Environment for Interactions. In *Proceedings of the fourth IEEE International Conference on Computer Vision Systems, New York (USA)*, 2006. (citation page 26)

[Ari 01] D. Arita and R.-I. Taniguchi. Rpv-ii : A stream-based real-time parallel vision system and its application to real-time volume reconstruction. In *Computer Vision Systems, Second International Workshop, ICVS, Vancouver (Canada)*, 2001. (citation page 26)

[Bau 74] B.G. Baumgart. *Geometric Modeling for Computer Vision.* PhD thesis, Stanford University, 1974. (citations pages 18, 19)

[Bli 82] J. Blinn. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3), 1982. (citation page 32)

[Bor 00] Eugene Borovikov and Larry Davis. A Distributed System for Real-time Volume Reconstruction. In *proceedings of CAMP-2000, IEEE*, 2000. (citation page 26)

[Boy 95] E. Boyer and M.O. Berger. 3D Surface Reconstruction Using Occluding Contours. In *Proceedings of 6th International Conference on Computer Analysis of Images and Patterns, Prague (Czech Republic)*, pages 198–205, September 1995. Lecture Notes in Computer Science, volume 970. (citation page 19)

[Boy 97] E. Boyer and M.-O. Berger. 3D Surface Reconstruction Using Occluding Contours. *International Journal of Computer Vision*, 22(3) : 219–233, 1997. (citation page 21)

[Boy 03] 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. (citations pages 5, 6, 8, 20, 21)

[Boy 06] E. Boyer. On Using Silhouettes for Camera Calibration. In *In Proceedings of the Seventh Asian Conference on Computer Vision, Hyderabad (India)*, 2006. (citations pages 11, 14, 16)

[Bra 04] 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, (USA)*, 2004. (citations pages 19, 21)

[Bro 01] 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)*, 2001. (citation page 24)

[Bro 04] G.-J. Brostow, I. Essa, D. Steedly, and V. Kwatra. Novel Skeletal Representation For Articulated Creatures. In *Proceedings of the 8th European Conference on Computer Vision, Prague, (Czech Republic)*, 2004. (citation page 34)

[Car 03a] J. Carranza, C. Theobalt, M Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *Proc. ACM Siggraph'03,* San Diego, USA, pages 569–577, July 2003. (citation page 32)

[Car 03b] J. Carranza, C. Theobalt, M. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. In *in Proc. of ACM SIGGRAPH, San Diego (USA)*, pages 569–577, 2003. (citation page 2)

[CGA] CGAL. the Computational Geometry Algorithms. http ://www.cgal.org. (citation page 19)

[Che 00] 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 2, pages 714 – 720, 2000. (citations pages 2, 26, 32)

[Chi 86] C.H. Chien and J.K. Aggarwal. Volume/surface octress for the representation of three-dimensional objects. *Computer Vision, Graphics and Image Processing*, 36(1) : 100–113, 1986. (citation page 18)

[Chu 03] C.-W. Chu, O. C. Jenkins, and M. J. Matarić. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, pages 475–482, 2003. (citation page 31)

[Cip 92] R. Cipolla and A. Blake. Surface Shape from the Deformation of Apparent Contours. *International Journal of Computer Vision*, 9 : 83–112, 1992. (citation page 19)

[Cip 95] R. Cipolla, K.E. Åström, and P.J. Giblin. Motion from the Frontier of Curved Surfaces. In *Proceedings of 5th International Conference on Computer Vision, Boston (USA)*, pages 269–275, June 1995. (citations pages 6, 11, 13)

[Cro 98] G. Cross and A. Zisserman. Quadric Reconstruction from Dual-Space Geometry. In *Proceedings of the 6th International Conference on Computer Vision, Bombay, (India)*, 1998. (citations pages 19, 21)

[Dem 77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *em* algorithm. *Journal of the Royal Statistical Society, series B (Methodological)*, 39(1) : 1–38, 1977. (citation page 37)

[Fau 01] O. Faugeras and Q.T. Luong. *The Geometry of Multiple Images*. MIT Press, 2001. (citation page 2)

[For 03] D. Forsyth and J. Ponce. *Computer Vision : A Modern Approach*. Prentice Hall, 2003. (citation page 2)

[Fra 03] J.-S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In *Proceedings of the 14th British Machine Vision Conference, Norwich, (UK)*, 2003. (citation page 23)

[Fra 04] J.-S. Franco, C. Menier, E. Boyer, and B. Raffin. A Distributed Approach for Real-Time 3D Modeling. In *IEEE CVPR Workshop on Real-Time 3D Sensors and their Applications, Washington DC*, July 2004. (citation page 26)

[Fra 05] 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)*, oct 2005. (citations pages 24, 25)

[Fur 04] Y. Furukawa, A. Sethi, J. Ponce, and D. J. Kriegman. Structure from Motion for Smooth Textureless Objects. In *Proceedings of the 8th European Conference on Computer Vision, Prague, (Czech Republic)*, 2004. (citations pages 11, 13)

[Fur 06] Y. Furukawa and J. Ponce. Carved Visual Hulls for Image-Based Modeling. In *Proceedings of the 9th European Conference on Computer Vision, Graz, (Austria)*, 2006. (citations pages 9, 133)

[Gav 96] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action : a multi-view approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, (USA)*, pages 73–80. IEEE Computer Society, 1996. (citation page 32)

[Gra 03a] 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)*, 2003. (citation page 24)

[Gra 03b] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3d structure with a statistical image-based shape model. In *Proceedings of the 9th International Conference on Computer Vision, Nice, (France)*, pages 641–648, 2003. (citation page 31)

[Gro 03] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Goll, S. Lang, K. Strehlke, A. Vande Moere, and O Staadt. blue-c : A spatially immersive display and 3d video portal for telepresence. In *In Proceedings of ACM SIGGRAPH*, 2003. (citations pages 2, 26)

[Har 00] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000. (citation page 2)

[Her 04] C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3) : 367–392, 2004. (citations pages 9, 133)

[Her 06] C. Hernandez, F. Schmitt, and R. Cipolla. Silhouette Coherence for Camera Calibratrion Under Circular Motion. *IEEE Transactions on PAMI*, 2006. to appear. (citations pages 12, 14, 15)

[Hil 04] A. Hilton, M. Kalkavouras, and G. Collins. Melies : 3d studio production of animated actor models. In *IEE European Conference on Visual Media Production*, 2004. (citation page 2)

[Isi 02] J. Isidoro and S. Sclaroff. Stochastic Mesh-Based Multiview Reconstruction . In *First International Symposium on 3D Data Processing, Visualization and Transmission, Padova (Italy)*, 2002. (citation page 19)

[Jos 95] T. Joshi, N. Ahuja, and J. Ponce. Structure and Motion Estimation from Dynamic Silhouettes under Perspective Projection. In *Proceedings of 5th International Conference on Computer Vision, Boston (USA)*, pages 290–295, June 1995. (citations pages 11, 13)

[Kam 00] Yoshinari Kameda, Takeo Taoda, and Michihiko Minoh. High Speed 3D Reconstruction by Spatio Temporal Division of Video Image Processing. *IEICE Transactions on Informations and Systems*, pages 1422–1428, 2000. (citation page 26)

[Kan 01] K. Kang, J.P. Tarel, R. Fishman, and D. Cooper. A Linear Dual-Space Approach to 3D Surface Reconstruction from Occluding Contours. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, (Canada)*, 2001. (citations pages 19, 21)

[Lap 06] M. Lapierre, J.-S. Franco, and E. Boyer. Visual shapes of Silhouette Sets. In *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Processing, Chapel Hill, (USA)*, 2006. (citations pages 5, 8, 10, 20)

[Lau 94] A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2) : 150–162, February 1994. (citations pages 5, 8, 18)

[Laz 01] 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, 2001. (citations pages 5, 7, 11)

[Li 02] M. Li, H. Schirmacher, M. Magnor, and H.-P. Seidel. Combining Stereo and Visual Hull Information for On-line Reconstruction and Rendering of Dynamic Scenes. In *In Proceedings of IEEE Multimedia Signal Processing MMSP, St. Thomas, US Virgin Islands*, 2002. (citation page 19)

[Lia 05] 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)*, 2005. (citations pages 19, 21)

[Luc 01] J. Luck, D.Small, and C. Q. Little. Real-Time Tracking of Articulated Human Models Using a 3D Shape-from-Silhouette Method. In *International Workshop RobVis*, pages 19–26, 2001. (citation page 34)

[Ḿ06] C. Ménier and E. Boyer. 3D Skeleton-Based Body Pose Recovery. In *Proceedings of the 3rd International Symposium on 3D Data Proces-*

sing, Visualization and Processing, Chapel Hill, (USA), 2006. (citations pages 32, 34)

[Mar 83] W.N. Martin and J.K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on PAMI*, 5(2) : 150–158, 1983. (citation page 18)

[Mar 98] 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*, 1998. (citation page 24)

[Mat 00] 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. (citation page 18)

[Mor 85] H. Moravec and A. Elfes. High-resolution maps from wide-angle sonar. In *IEEE International Conference on Robotics and Automation*, 1985. (citation page 24)

[Nar 98] P.J. Narayanan, P.W. 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. (citations pages 2, 26)

[Nie 94] W. Niem. Automatic Modelling 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. (citation page 18)

[Nis 05] M. Niskanen, E. Boyer, and R. Horaud. Articulated motion capture from 3-d points and normals. In *British Machine Vision Conference*, volume 1, pages 439–448, September 2005. (citation page 32)

[Pla 03] R. Plaenkers and P. Fua. Articulated Soft Objects for Multi-View Shape and Motion Capture. *IEEE Transactions on PAMI*, 25(10), 2003. (citations pages 32, 33)

[Sen 03] A. Senior. Real-time articulated human body tracking using silhouette information. In *5th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, october 2003. (citation page 32)

[Ser 82] J. Serra. *Image Analysis and Mathematical Morphology, Volume I.* Academic Press, 1982. (citation page 37)

[Sin 04] S.N. Sinha, M. Pollefeys, and L. McMillan. Camera Network Calibration from Dynamic Silhouettes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington, (USA)*, 2004. (citations pages 11, 13)

[Sin 05] S. Sinha and M. Pollefeys. Multi-view Reconstruction using Photo-consistency and Exact Silhouette Constraints : A Maximum-Flow Formulation. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, (China)*, 2005. (citations pages 9, 133)

[Smi 01] C. Sminchisescu and B. Triggs. Covariance Scaled Sampling for Monocular 3D Human Tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Kauai, (USA)*, 2001. (citation page 32)

[Sno 00] 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)*, 2000. (citation page 24)

[Sta 03] J. Starck and A. Hilton. Model-based multiple view reconstruction of people. In *Proceedings of the 9th International Conference on Computer Vision, Nice, (France)*, pages 915–922, 2003. (citation page 2)

[Sul 98] 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. (citation page 19)

[Sze 93] R. Szeliski. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 58(1) : 23–32, 1993. (citation page 18)

[The 02] C. Theobalt, M. Magnor, P. Schüler, and H.-P. Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. *International Journal of Image and Graphics*, 4(4) : 563–584, October 2002. Pacific Graphics 2002. (citation page 34)

[Vai 92] R. Vaillant and O. Faugeras. Using Extremal Boundaries for 3-D Object Modeling. *IEEE Transactions on PAMI*, 14(2) : 157–173, February 1992. (citation page 19)

[W0̈4] S. Würmlin, E. Lamboray, and M. Gross. 3D Video Fragments : Dynamic Point Samples for Real-time Free-Viewpoint Video. *Computers & Graphics, Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data*, 28(1) : 3–14, 2004. (citations pages 2, 26)

[Wu 03] X. Wu and T. Matsuyama. Real-Time Active 3D Shape Reconstruction for 3D Video. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, Rome (Italy)*, pages 186–191, September 2003. (citation page 26)

[Zen 04] 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)*, 2004. (citation page 24)