HAL
open science

Système de vision temps-réel pour les interactions

Clement Menier

To cite this version:

Clement Menier. Système de vision temps-réel pour les interactions. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 2007. Français. NNT : . tel-00379206

HAL Id: tel-00379206

https://theses.hal.science/tel-00379206

Submitted on 28 Apr 2009

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.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : "Informatique : Images, Vision et Robotique"

préparée au laboratoire GRAVIR
dans le cadre de ***l'École Doctorale "Mathématiques, Sciences et Technologies de l'Information, Informatique"***

par

Clément MÉNIER

Système de vision temps-réel pour les interactions

Directeur de thèse : M. Radu Horaud

JURY

M. JAMES CROWLEY	, Président
M. PASCAL GUITTON	, Rapporteur
M. ADRIAN HILTON	, Rapporteur
M. RADU HORAUD	, Directeur de thèse
M. EDMOND BOYER	, Co-encadrant
M. BRUNO RAFFIN	, Co-encadrant

Remerciements

Je tiens tout d'abord à remercier Edmond Boyer et Bruno Raffin pour leur encadrement lors de ces travaux et pour leus efforts qui ont permis la réalisation de la plateforme expérimentale GrImage sans laquelle ces travaux n'auraient pas été possibles.

Durant cette thèse, j'ai eu le plaisir de collaborer avec de nombreuses personnes. Je remercie Jean-Sébastien Franco, Luciano Soares qui m'ont permis d'avancer plus vite dans mes travaux, Florence Bertails qui m'a fait découvrir de nouveaux domaines et Florian Geffray qui a grandement participé à l'élaboration et à la diffusion de ces travaux lors de démonstrations publiques. Je tiens à remercier plus particulièrement Jérémie Allard pour son aide constant et sa motivation à effectuer des démonstrations toujours plus audacieuses, jusqu'à y passer des nuits complètes.

Je remercie aussi Hervé Mathieu, Nicolas Turro et Jean-François Cuniberto pour l'aide technique précieuse qu'ils m'ont rendue pour que la plateforme GrImage soit opérationnelle.

Merci aux membres de mon jury, James Crowley, Pascal Guiton et Adrian Hilton, d'avoir accepté d'en faire partie. Les échanges avec eux m'ont permis de découvrir des points de vue différents sur ce domaine.

Je remercie aussi les équipes MOAIS et PERCEPTION pour leur accueil et les discussions enrichissantes.

Enfin je tiens à remercier ma famille qui m'a soutenu moralement pendant ces trois ans. Je remercie particulièrement ma femme Domitille qui m'a encouragé durant toute cette période et à qui je dédie cette thèse.

Table des matières

1 Introduction **1**
1.1 Contexte 1
1.2 Problématique 2
1.3 Contributions 3
1.4 Organisation 5

I Système de Vision **7**

2 Système de Vision **9**
2.1 Etat de l'Art des systèmes d'interaction 9
2.1.1 Système physique 10
2.1.2 Systèmes sans marqueurs 12
2.2 Conception du système 16
2.2.1 Formalisation de la chaîne de traitement 16
2.2.2 Les différentes étapes 17
2.2.3 Stratégies de distribution 21
2.3 Bilan 25

II Modélisation des formes **27**

3 État de l'art **29**
3.1 Catégories de modélisation des formes 29
3.1.1 Les méthodes actives 29
3.1.2 Les méthodes passives 31
3.2 Méthodes de modélisation d'enveloppes visuelles 32
3.2.1 Les approches volumiques 33
3.2.2 Surfaciques 33
3.2.3 Hybrides 35
3.3 Bilan 35

4 Modélisations surfaciques distribuées **37**
4.1 Introduction 37
4.2 Calcul des segments de vue 39
4.3 Modélisation par tétraédrisation 41
4.4 Modélisation surfacique 43
4.5 Résultats 46
4.6 Bilan 48

5 Octree parallélisé **49**
5.1 Modélisation par octree 50
5.2 Parallélisation 50
5.2.1 Enjeux 50
5.2.2 Principe du parallélisme adaptatif par vol de tâches 52
5.2.3 Adéquation avec l'octree 52
5.2.4 Contrainte relâchée de parcours en largeur 53
5.3 Modélisation volumique distribuée : détails de l'implantation 53
5.4 Résultats 54
5.4.1 Exécution hors-ligne 54
5.4.2 Exécution temps réel 58
5.5 Bilan 60

III Mise en œuvre **63**

6 Implantation **65**
6.1 Le modèle FlowVR 66
6.2 Implantation 69
6.2.1 Implantation modulaire 69
6.2.2 Filtrage et schéma de communications collectives 69
6.2.3 Gestion de l'asynchronisme 71
6.3 Acquisition et traitements des images 72
6.3.1 Acquisition synchrone des images 73
6.3.2 Extraction des silhouettes 73
6.3.3 Vectorisation des silhouettes 74
6.4 Bilan 75

7 Applications **77**
7.1 La plateforme GrImage 77
7.2 Applications interactives 78
7.2.1 Intégration visuelle 78
7.2.2 Interactions avec des objets rigides 79
7.2.3 Perruque virtuelle 80

7.2.4 Sculpture 81
7.2.5 Interactions avec un fluide 82
7.3 Performances 82
7.3.1 Débit et latence 83
7.3.2 Précision 83
7.4 Bilan 84

IV Modélisation du mouvement 87

8 État de l'art 89
8.1 Approches par apprentissage 90
8.2 Approches par ajustement de modèle 91
8.3 Approches sans *a priori* 92
8.4 Bilan 93

9 Principe 95
9.1 Modèle articulé filaire 95
9.2 Données d'observation squelettiques 96
9.3 Suivi du modèle 98
9.3.1 Modèle génératif probabiliste 98
9.3.2 Ajustement 101
9.3.3 Suivi au cours d'une séquence 101
9.4 Bilan 102

10 Tests 105
10.1 Séquences de test 105
10.2 Résultats 106
10.3 Discussion 110
10.3.1 Robustesse 110
10.3.2 Performance 111
10.4 Bilan 112

V Conclusion et perspectives 113

11 Conclusion et perspectives 115
11.1 Bilan global sur nos travaux 115
11.2 Parallélisme et interactions : un bon tandem 116
11.3 Les applications envisageables et leurs besoins 117
11.4 Vers les interactions Homme-Machine 118

Table des notations

N	Nombre de caméras
L	Latence
D	Débit
$d = \frac{1}{D}$	délai entre deux trames consécutives
E_i	Étape i du pipeline
t_i	Temps d'exécution séquentiel pour une trame de l'étape E_i
n_i	Nombre d'unités de calcul pour l'étape E_i
p_i	Nombre de processeurs alloués à la parallélisation par trame de l'étape E_i
	Degré de parallélisation des unités de calcul de E_i
X_i	Ensemble des points composant les données squelette extraites
S	Pose de l'utilisateur
a_i	Segment associé au point X_i

Introduction 1

1.1 Contexte

Depuis quelques années, la traditionnelle $2D$ laisse de plus en plus place à la $3D$ dans le monde informatique. Ce basculement fut amorcé par la démocratisation des cartes graphiques actuelles qui ont facilité la conception d'applications $3D$ et dont les performances ont permis de rendre des effets visuels de grande qualité en particulier dans les jeux vidéo. Plus récemment, cette révolution $3D$ s'est emparée des interfaces graphiques des grands systèmes d'exploitation : Aero pour Windows Vista, XGL pour Unix et Aqua pour Mac OS X. Ce basculement est d'autant plus complet que certains constructeurs comme Sharp ou Philipps proposent désormais des écrans LCD $3D$ auto-stéréoscopiques de série permettant de visualiser ces effets $3D$ naturellement. De grands progrès ont aussi été réalisés dans les casques de réalité virtuelle. La partie capteur a elle aussi subi des modifications avec l'apparition de souris gyroscopiques (navigation dans l'espace) ou de manettes captant les mouvements naturels, grâce à des accéléromètres par exemple, comme celle utilisée par la nouvelle console de jeu Wii de Nintendo. La suite naturelle de cette évolution est de se poser la question : comment peut-on acquérir des informations $3D$ sur l'ensemble de l'utilisateur tout en se passant de ces manettes ?

Une réponse à cette question vient de la communauté de la vision par ordinateur. Les récents progrès dans ce domaine permettent en effet d'obtenir de manière assez robuste un grand nombre d'informations $3D$ sur une scène filmée par plusieurs caméras. Ces techniques présentent l'avantage pour l'utilisateur de ne pas nécessiter de porter un équipement spécial, rendant les interfaces Homme-Machine les plus naturelles possibles.

Pour obtenir ces données en temps réel, les approches actuelles n'extraient que des informations relativement simples ou peu précises.

Pour repousser ces limitations dues aux coûts importants des calculs mis en jeu, la distribution de ces calculs sur plusieurs processeurs est nécessaire. En effet la croissance de la puissance des ordinateurs n'est désormais que très peu générée par l'augmentation de la puissance des processeurs mais plutôt par la multiplication de ceux-ci au sein d'une même machine. De plus en plus fréquemment les ordinateurs actuels disposent de plusieurs processeurs chacun comportant plusieurs coeurs. Les cartes graphiques actuelles offrent aussi pour certains types de calcul dédiés des processeurs parallèles. Depuis peu, NVIDIA propose une architecture, baptisée G80, comprenant jusqu'à 128 unités de traitement accessibles à des calculs plus généraux (GPGPU : General-Purpose GPU).

Modéliser, en $3D$ et avec précision, un utilisateur en temps réel offre de nombreuses possibilités d'applications. Tout d'abord, la possibilité de transmettre une image tridensionnelle de la personne permet d'envisager des systèmes de visioconférence $3D$, augmentant d'autant le réalisme de la communication et améliorant la sensation de présence et de proximité de l'interlocuteur. L'industrie cinématographique est aussi intéressée par l'obtention d'informations $3D$ sur les acteurs et plus généralement sur la scène pour réaliser de nouveaux effets visuels. A plus long terme des films $3D$ laissant à l'utilisateur la liberté du choix du point de vue de la caméra pourront être proposés. Enfin dans le domaine de la réalité virtuelle un tel système peut permettre à l'utilisateur d'interagir directement avec les objets virtuels le plus naturellement possible.

1.2 Problématique

Dans cette thèse nous cherchons à obtenir en temps réel des informations $3D$ sur une scène et plus particulièrement sur un utilisateur interagissant avec une application finale. Ces données sont extraites à partir d'images de cette scène filmées depuis plusieurs caméras. Cette problématique comporte certaines difficultés de par les contraintes liées aux interactions.

Les systèmes de vision permettant d'extraire des informations $3D$ telles que la géométrie de l'utilisateur sont généralement complexes. L'ensemble des images prises à un instant, baptisé *trame*, doit subir successivement de nombreux traitements pour obtenir le résultat final. Selon les paradigmes d'interaction mis en application, différents types de données (géométrie, suivi de régions, identification de la tête, ...) doivent être extraits avec des critères de qualité et de performances qui varient. Chaque type d'information nécessite des traitements spécifiques ce qui complexifie d'autant le système. Un

grand soin doit donc être apporté à la conception du système pour factoriser les étapes communes et le rendre suffisemment flexible pour s'adapter aux besoins des applications finales. Le système doit aussi gérer le fait que les ressources telles que les caméras soient distribuées et non centralisées sur une seule machine : les ordinateurs actuels ne peuvent généralement pas gérer plus de 2 ou 3 caméras à cause des limites matérielles telles que le bus gérant les entrées/sorties.

Une autre difficulté concerne le choix des approches d'extraction de données $3D$. Celui-ci doit non seulement prendre en compte la qualité de ces données mais aussi les temps de calcul pour les obtenir. La plupart des travaux existants de modélisation $3D$ à partir d'images cherchent à obtenir l'information de la meilleure précision possible mais ne prennent pas en compte en sus les critères de temps de calcul. Deux mesures sont particulièrement importantes : le *débit* qui correspond au nombre de trames traitées par seconde et la *latence* qui correspond au temps de traitement d'une trame, c'est-à-dire au délai entre la capture des images et l'obtention du résultat. Ce problème multi-critère est particulièrement délicat étant donné que les paradigmes d'interaction nécessitent des rapports qualité/performance qui peuvent variés entre eux. De plus il est important de choisir des techniques nécessitant le minimum d'*a priori* sur la scène ou sur l'utilisateur pour que le système puisse être déployé partout et utilisé par le plus grand nombre.

Enfin les algorithmes de vision par ordinateur sont très gourmands en ressources de calcul. Pour obtenir des performances satisfaisant les contraintes liées aux interactions en particulier concernant la latence, les calculs de chaque traitement doivent être parallélisés. Cette distribution des calculs est particulièrement délicate du fait que le temps de traitement d'une trame par chaque étape est très cours (de l'ordre de la dizaine de millisecondes).

1.3 Contributions

Cette thèse présente une chaîne complète d'extraction d'informations $3D$ sur une scène à partir de caméras. Ces travaux concernent différents aspects de cette chaîne.

Un système de vision

Partant d'une étude des différents types d'informations $3D$ utiles pour les interactions et des approches permettant de les calculer, nous proposons une formalisation de la chaîne d'acquisition de ces données par un découpage fonctionnel. Les étapes communes se retrouvent factorisées et nous aboutissons à un pipeline comportant plusieurs niveaux : acquisition synchrone des images, extraction des silhouettes, modélisation de la géométrie tri-dimensionnelle de l'utilisateur puis de ses mouvements. Ce découpage fonctionnel permet de choisir pour chaque application le ou les meilleurs algorithmes pour chaque

étape selon les critères de qualité, performance et types d'informations souhaitées.

Nous démontrons la flexibilité et l'adéquation de ce système de vision avec les paradigmes d'interaction en l'implantant pour différentes applications. En utilisant l'intergiciel FlowVR, un soin particulier a été porté sur cette mise en oeuvre pour conserver la modularité du système et gérer au mieux la distribution des ressources. L'intérêt de ce système a été validé lors de démonstrations grand public. Ces tests effectués dans des endroits très variés et cummulant plusieurs centaines d'heures permettent aussi de rendre compte de la reproductibilité, stabilité et robustesse du système proposé.

Parallélisation de modélisations de formes

L'objectif étant d'obtenir les données $3D$ en temps réel, nous avons étudié la parallélisation d'algorithmes de modélisation de la géométrie de la scène. De manière générique, nous mettons en avant les différentes stratégies de répartition des calculs dont nous disposons et montrons leurs impacts sur les performances du système. Une première forme de répartition des calculs se présente naturellement sous la forme d'un pipeline distribué. Une seconde forme, spécifique à chaque algorithme, consiste à paralléliser les calculs nécessaires pour traiter une trame. Nous proposons une telle parallélisation pour plusieurs algorithmes de modélisation des formes : un algorithme surfacique permettant d'obtenir une surface précise de la scène et un algorithme volumique permettant d'obtenir l'occupation de l'espace sous la forme d'une grille hiérarchique. La parallélisation permet d'améliorer de façon très significative la réactivité (latence) et la fréquence de traitement (débit) du système.

Modélisation du mouvement

Pour obtenir des informations de suivi ou d'identification des parties du corps de l'utilisateur, nous proposons une approche originale de modélisation du mouvement. Celle-ci diffère des autres techniques de suivi du mouvement de par le fait qu'elle cherche à minimiser les *a priori* sur l'utilisateur tout en conservant une approche de modèle articulé. Cette méthode s'adapte facilement à n'importe quel utilisateur et est donc particulièrement adaptée aux systèmes interactifs. Sa portée est toutefois plus générale car elle offre une séparation claire entre le problème de la modélisation des formes et de celle du mouvement.

Valorisation

Ces travaux ont fait l'objet de plusieurs publications scientifiques. Les modélisations des formes distribuées ont été publiées au Workshop Real-Time Vision lors de CVPR'04 [Franco 04], au symposium IPT'04 [Ménier 04], à la conférence ORASIS'04 [Ménier 05] et en poster à IEEEVR'07 [Soares 07]. La mise en oeuvre du système a été présenté au symposium IPT'05 [Allard 05] et à la conférence ICVS'06 [Allard 06]. Enfin la modélisation du mouvement est décrite dans un papier présenté lors de la conférence 3DPVT'06 [Ménier 06].

Ces travaux ont aussi fait l'objet de très nombreuses démonstrations. Lors de manifestations grand public comme la fête de la Science ou des journées portes ouvertes, le système a recueilli un immense succès autant auprès des adultes que des enfants. Celui-ci a aussi été présenté à un public plus spécialisé lors de démonstrations internes ou lors de salons professionnels tels que Vision'06.

Enfin ces travaux font l'objet actuellement d'une valorisation industrielle par un transfert technologique.

1.4 Organisation

La partie I s'intéresse aux systèmes de vision pour les interactions dans leur ensemble. En s'appuyant sur un état de l'art des principaux travaux de système d'interaction et en synthétisant ces approches, nous décrivons une conceptualisation de la chaîne de traitement en étapes fonctionnelles. Nous proposons dans ce chapitre 2 un système allant de l'acquisition des images au rendu graphique. Nous présentons aussi les différentes stratégies de distribution et leurs impacts sur les performances.

La partie II s'intéresse à la modélisation des formes, c'est-à-dire aux algorithmes d'extraction de la géométrie de l'utilisateur à partir des images. Le chapitre 3 présente les grandes familles d'approches existantes en vision par ordinateur. Cet état de l'art se focalise plus particulièrement sur le modèle d'enveloppe visuelle, modèle bien adapté aux applications interactives car fournissant un excellent rapport entre qualité et performances. La parallélisation de différentes méthodes de calcul d'enveloppes visuelles est alors décrite. Le chapitre 4 présente la parallélisation d'un algorithme de calcul de la surface de l'enveloppe visuelle polyédrique exacte ou approximée. Le chapitre 5 présente la parallélisation d'un algorithme de modélisation par octree basée sur des approches adaptatives.

La partie III présente la mise en oeuvre du système avec les modélisations des formes présentées. Le chapitre 6 décrit l'implantation avec l'outil FlowVR et les détails de la gestion du réseau et de la cohérence des données. Plusieurs applications expérimentales ont été réalisées et sont décrites dans le chapitre 7. Différentes interactions ont ainsi été testées et démontrent les atouts du système proposé.

La partie IV présente la modélisation du mouvement, c'est-à-dire le suivi du mouvement de l'utilisateur à partir d'un modèle articulé. Le chapitre 8 présente un état de l'art sur les principales méthodes de suivi du corps humain. Nous présentons ensuite dans le chapitre 9 le principe de notre approche. Cette méthode sépare clairement le mouvement de la forme de l'utilisateur tout en gardant un modèle articulé filaire de celui-ci. Des résultats expérimentaux sont présentés dans le chapitre 10 pour montrer l'intérêt et les limites de cette méthode.

Enfin nous concluons dans la partie V ces travaux et nous nous attardons sur les perspectives qu'ils ouvrent.

I

Système de Vision

Système de Vision 2

Pour permettre à un utilisateur d'interagir avec une application, il est nécessaire de fournir à cette dernière des informations sur l'utilisateur. Nous appellerons *système d'interaction* un ensemble regroupant un ou plusieurs capteurs physiques avec une architecture logicielle et matérielle transformant les données brutes en informations utiles pour les applications finales. Remarquons que nous ne nous intéressons pas au paradigme d'interaction, c'est-à-dire comment ses informations sont utilisées par les applications, mais uniquement à la manière d'obtenir ses informations.

2.1 Etat de l'Art des systèmes d'interaction

Plusieurs systèmes d'interaction ont été proposés dans les domaines des interactions Homme-Machine ou de la vision par ordinateur. Nous nous basons sur les quatre critères suivants pour distinguer ces différents systèmes :

Informations extraites : la nature et la richesse des informations extraites conditionnent très fortement le type d'interactions finales réalisables. Le type d'informations extraites est très lié au capteur matériel utilisé. Voici quelques grandes catégories de données utiles :

- les **primitives images** correspondent à des zones particulières identifiées dans l'image : la *silhouette* de la zone d'intérêt (utilisateur + objets manipulés), le visage ou les régions de peau.
- la **position** dans l'espace de points liés à l'utilisateur.
- la **forme** de l'utilisateur correspond à sa géométrie et à son positionnement dans l'espace.

- le **suivi** fournit un lien entre des informations de trames consécutives.
- l'**identification** apporte une information de plus haut niveau sur le "sens" des données (mains, pied, tête, ...).
- la **photométrie** correspond à l'information nécessaire pour une visualisation selon un nouveau point de vue de la scène.

Précision : la précision des données joue un rôle crucial dans la finesse des interactions réalisées. Une précision inférieure au centimètre est par exemple requise pour pouvoir interagir avec les doigts séparément.

Performances : les performances du système en terme de débit et de latence ont un grand impact sur la qualité des interactions. En particulier la stabilité des simulations interactives décroit avec le déplacement effectué par l'utilisateur entre deux trames. Plus les mouvements de ce dernier sont rapides, plus le débit doit être important. Plus la latence est faible, plus le système est réactif. Une latence inférieure à 100 ms est requise pour que l'utilisateur ne soit pas trop perturbé.

Usages et limites : les contraintes techniques peuvent limiter l'usage du système à un nombre restreint d'utilisateurs, par exemple lorsque de forts *a priori* sur celui-ci sont nécessaires. Ces contraintes peuvent aussi limiter les mouvements autorisés ou bien perturber l'utilisateur en le gênant. Idéalement, le système devrait pouvoir être utilisé par n'importe qui et laisser l'utilisateur interagir le plus naturellement possible.

Pour permettre une comparaison entre les méthodes nous considérerons par la suite un espace d'interaction de $2\mathrm{m} \times 2\mathrm{m} \times 2\mathrm{m}$.

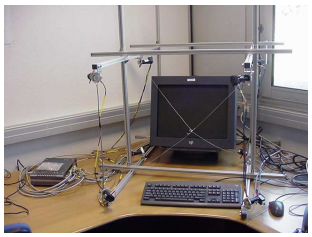
2.1.1 Système physique

Une première catégorie de systèmes obtient des informations sur l'utilisateur en mesurant la position et/ou l'orientation d'objets physiques, appelés marqueurs ou cibles, tenus ou attachés à cet utilisateur. Les informations obtenues sont alors la position et le suivi de ces points. L'identification de ces points est statique et correspond à l'endroit du corps où ils sont attachés. Différentes solutions technologiques ont été proposées, certaines étant déjà commercialisées.

Les systèmes éléctromagnétiques, comme le Flock of Birds ® [Fob], mesurent la position et l'orientation des marqueurs à l'aide d'ondes et de champs magnétiques générés. Ces systèmes sont cependant fortement perturbés par les masses métalliques avoisinantes [Milne 96]. D'autres solutions analogues ont été proposées en utilisant des ondes sonores [Reunanen 95] ou des capteurs inertiels [Intersense , Xsens] ou l'association de plusieurs de ces approches.

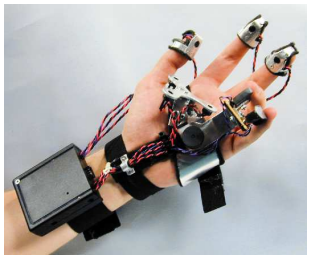
Différents systèmes optiques ont aussi été proposés et sont désormais commercialisés. Le système Optotrak ®utilise des projecteurs et des photo-récepteurs nécessitant

(a) Phantom



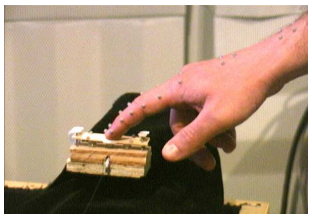
(b) Spidar

(c) Capteur inertiel



(d)

(e) Vicon



(f)

Figure 2.1 *Différents systèmes d'interaction utilisant des capteurs ponctuels.*

des liaisons éléctroniques entre l'utilisateur et le système. Les systèmes Vicon ®ou Arttrack ®utilisent des marqueurs réfléchissants avec des flashs et caméras infra-rouge pour obtenir la position de ces cibles. En utilisant un grand nombre de cibles sur une main, ce type de système a récemment permis d'obtenir des interactions d'une grande précision [Kry 06]. L'utilisation de plusieurs marqueurs, combinée à la connaissance d'un modèle *a priori* de l'utilisateur, permet d'obtenir une approximation de sa forme [Carranza 03]. Cette géométrie est une information extrapolée et non mesurée. Elle est donc relativement imprécise ou requiert un modèle surfacique de l'utilisateur de très grande qualité.

Enfin les systèmes mécaniques obtiennent la position de ces cibles en les reliant physiquement à une base fixe. Le Phantom ® [Massie 94] (voir figure 2.1(a)) utilise pour cela un bras articulé. Le Spidar [Ishii 94] (voir figure 2.1(b)) repose quand à lui sur la mesure de distances à l'aide de fils. L'intérêt de ces systèmes réside dans la possibilité de donner à l'utilisateur un retour haptique. En revanche un seul et unique point peut être mesuré. Des systèmes mécaniques basés sur un exosquelette permettent également de connaître précisément la position de chaque membre de l'utilisateur donnant une information plus globale. Les mouvements de l'utilisateur restent cependant très contraints par l'encombrement de ce type de système.

Les systèmes à base de marqueurs permettent donc d'obtenir avec une grande précision et une fréquence importante la position et l'orientation de plusieurs points de référence placés sur l'utilisateur. Ces systèmes sont en revanche assez lourds car ils nécessitent d'équiper l'utilisateur. Ceci impose un grand temps de préparation pour la pose de ces marqueurs interdisant un changement fréquent d'utilisateurs. De plus ces marqueurs peuvent gêner physiquement les mouvements de l'utilisateur rendant l'interaction désagréable.

2.1.2 Systèmes sans marqueurs

Pour pallier aux problèmes liés aux marqueurs, des travaux se sont intéressés à l'obtention d'informations directement grâce à des caméras à l'aide de techniques de vision par ordinateur. La difficulté est alors d'identifier les informations dans les images et d'en extraire les données utiles pour les interactions.

2.1.2.1 Systèmes mono-caméra

Une catégorie de systèmes d'interaction sans marqueurs repose sur l'utilisation d'une seule caméra pour recueillir des primitives $2D$ sur l'utilisateur dans l'image. Ces systèmes mono-caméra ont particulièrement été étudiés pour les interactions dans les domaines de la réalité augmentée ou mixte et des interactions Homme-Machine.

Le premier système interactif mono-caméra, VideoPlace, a été développé par l'équipe de Myron Krueger dans les années 1975 [Krueger 85]. Une architecture matérielle dédiée permet d'extraire en temps réel la silhouette de l'utilisateur. Cette dernière permet alors de contrôler diverses applications simples comme du dessin avec la main (voir figure 2.2). Ces travaux ont inspiré par la suite un très grand nombre d'autres systèmes d'interaction. Les fortes capacités de calcul des processeurs actuels permettent à ces nouveaux systèmes de ne plus nécessiter une architecture matérielle dédiée et de réaliser des tâches plus complexes telles que la détection et le suivi d'objets particuliers dans les images : doigts [Crowley 95], visage [Zelinsky 96, Darrel 95] ou regard [O'Hagan 97]. La maturité de ces systèmes et leur faible coût permet à l'heure actuelle leur commercialisation auprès du grand public. L'exemple le plus frappant est très certainement l'eyetoy ® développé pour la station de jeux SONY.

Les informations obtenues par ces systèmes sont des primitives images $2D$ les rendant facilement utilisables dans le cadre des interfaces d'interaction $2D$ actuelles. Cependant elles s'avèrent très limitées lorsque l'on considère des interactions dans un univers tridimensionel tel que Workbench ou Cave.

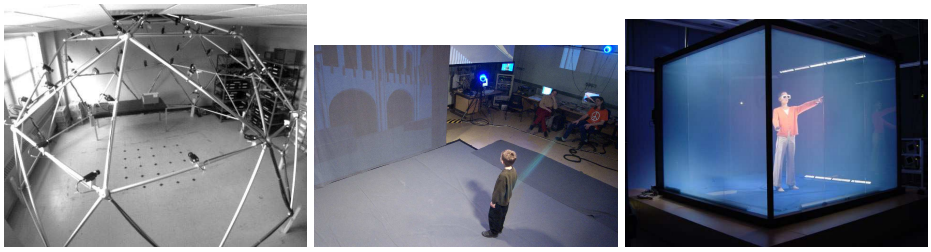
Figure 2.2 *VideoPlace : premier système d'interaction mono-caméra proposé par Myron Krueger.*

2.1.2.2 Système Multi-caméra

Pour obtenir des informations 3D complètes sur la scène, celle-ci doit être observée depuis plusieurs points de vue simultanément. Un tel système multi-caméra ne peut cependant pas être construit par simple juxtaposition de systèmes mono-caméra. En effet des problèmes spécifiques apparaissent : acquisition et traitement synchrone des images, gestion du flux important des informations brutes (images couleurs), extraction des informations 3D coûteuse.

La plupart des systèmes multi-caméras se sont intéressés à l'extraction de la géométrie de l'utilisateur. Cette information tri-dimensionnelle est très riche car elle permet d'obtenir le positionnement global de l'utilisateur.

Le premier système multi-caméra *Virtualized Reality* fut proposé par Kanade *et al.* [Kanade 97] (voir figure 2.3(a)). Il est composé d'un dôme comportant 49 caméras. L'acquisition des images est réalisée dans un premier temps sur plusieurs disques de manière synchrone grâce à un système électronique. Ces données sont ultérieurement traitées pour obtenir un modèle $3D$ de l'utilisateur pour chaque trame. Ce système n'est pas interactif mais *hors-ligne* : aucune contrainte de performances n'est donc imposée



(a) Virtualized Reality (b) BBC (c) Blue-C

Figure 2.3 *Différents systèmes multi-caméras.*

à ce type de systèmes. D'autres systèmes hors-lignes ont été proposés par Hilton *et al.* [Hilton 04], par la BBC [Graux 04] (voir figure 2.3(b)). Ces systèmes proposent des améliorations sur la qualité des modèles obtenus en utilisant des algorithmes plus avancés. Ils n'apportent cependant pas de réponse sur le caractère non-interactif de ces systèmes.

Plus récemment des sytèmes interactifs (*en-ligne*) ont été proposés. Cheung *et al.* [Cheung 00] propose un tel système d'interaction qui permet d'obtenir un modèle $3D$ de l'occupation de l'utilisateur dans l'espace et des informations sur ses mouvements. Pour répondre à la contrainte du temps réel imposée par l'interactivité du système, ce système se limite à des calculs approximatifs et simples du modèle $3D$ (précision de l'ordre de 3 cm) et n'utilise que 5 caméras. L'utilité de ces approches pour les interactions a été démontrée par Hasenfratz *et al.* [Hasenfratz 04] qui propose une approche identique utilisant la carte graphique.

Pour gérer plus de points de vue, Davis *et al.* [Borovikov 03] proposent une solution distribuée dans un contexte hors-ligne (les images sont lues à partir d'une base de données distribuée). Une modélisation des formes à 10 trames par seconde et avec une précision de $1,5$ cm a ainsi pu être obtenue avec 14 points de vue et 16 PCs. Matsuyama *et al.* [Wu 06] proposent une solution distribuée complète : l'acquisition et les traitements de vision étant répartis sur l'ensemble des machines disponibles. L'inconvénient de ces méthodes est leur complexité cubique en la précision liée à leur algorithme de modélisation. Ainsi pour doubler la précision il faut 8 fois plus de calculs. Pour obtenir une modélisation à 30 trames par seconde et avec une précision inférieure au centimètre, il est nécessaire d'utiliser au moins une centaine de processeurs.

Le projet Tele-immersion 3D [Kelshikar 03], fruit de la collaboration entre les universités de Pennsylvanie (Upenn) et de Caroline du Nord (UNC), a permis la démonstration lors de la manifestation SuperComputing 2002 de la faisabilité d'une modélisation 3D précise en temps réel. Les besoins en puissance de calculs sont cependant tels qu'il est nécessaire d'effectuer ceux-ci sur plus d'une centaine de machines situées à distance

entraînant une forte latence.

D'autres systèmes ont été proposés s'intéressant plus particulièrement à la génération d'une image de la personne selon un nouveau point de vue (information photométrique). Le système Blue-C [Gross 03] de l'université de Zurich (voir figure 2.3(c)) permet d'obtenir un modèle graphique de l'utilisateur en temps réel. Un nuage de points colorés correspondant à la surface de l'utilisateur est ainsi calculé. Ce type de modèle permet un affichage selon un point de vue virtuel, mais est assez difficile à utiliser dans des applications interactives : en particulier aucun maillage n'est fourni. De plus pour permettre une exécution temps réel, l'hypothèse que l'utilisateur bouge relativement peu entre deux trames consécutives est nécessaire et les données sont assez peu précises.

Li *et al.* [Li 03, Li 04] proposent un système utilisant les capacités des cartes graphiques modernes (GPU) pour générer une image vituelle selon un nouveau point de vue. Ils utilisent pour cela une technique similaire à [Matusik 00] nécessitant la connaissance du point de vue de visualisation. L'inconvénient principal des approches générant seulement un nouveau point de vue consiste en le fait qu'aucun véritable modèle tri-dimensionnel de l'utilisateur n'est disponible rendant ces techniques uniquement utilisables pour des interactions visuelles.

Certains systèmes utilisent des approches baptisées *lumières structurées* reposant sur la projection d'une figure lumineuse par un projecteur et en sa détection dans les images filmées [Waschbüsch 05]. Ces techniques sont toutefois gênantes pour l'utilisateur, l'éblouissant lorsqu'il regarde le projecteur. De plus, en raison des contraintes matérielles, il est encore difficile d'obtenir une modélisation complète de la personne, 3 ou 4 caméras/projecteurs maximum pouvant être utilisés.

Enfin notons que certains systèmes s'intéressent aussi au suivi et à l'identification des mouvements de l'utilisateur comme Cheung *et al.* [Cheung 00] ou De La Rivière *et al.* [delaRivière 05]. Carranza *et al.* [Carranza 03] proposent un système hors-ligne permettant de suivre les mouvements d'une personne grâce à un modèle articulé de grande précision. Ils proposent de plus de retrouver la forme de l'utilisateur à partir de ces informations et du modèle articulé. Cependant tout comme dans les systèmes avec plusieurs marqueurs, ces modèles géométriques sont inexacts car non mesurés par rapport à la réalité.

2.2 Conception du système

La conception d'un système d'interaction multi-caméra repose sur le développement d'une chaîne complète de traitements allant de l'acquisition des images au rendu final en passant par l'extraction d'informations sur l'utilisateur. L'élaboration d'un tel système pose certaines difficultés.

Premièrement celui-ci doit être suffisamment flexible et générique pour s'adapter aux différents besoins des applications finales. Selon les paradigmes d'interaction mis en jeu, divers types d'informations doivent être extraites avec des contraintes de qualité, de performance ou d'usage spécifiques. Deuxièmement, pour chaque traitement une recherche des algorithmes offrant les meilleurs compromis entre qualité et performances doit être effectuée en prenant en compte l'adéquation des résultats avec les besoins des applications finales ou des autres traitements qui en dépendent. Finalement un grand soin doit être apporté à l'efficacité du système malgré sa complexité. En particulier il est nécessaire de gérer au mieux la distribution des calculs sur l'ensemble des ressources disponibles. La parallélisation des traitements impliqués dans un système multi-caméra est particulière de par son caractère multi-critère (débit et latence) et nécessite des réponses spécifiques.

Nous proposons un système d'interaction multi-caméra répondant à ces problèmes. Ce système est basé sur une formalisation de la chaîne de traitement en une série d'étapes fonctionnelles. Ce découpage conceptuel permet de garantir au système la flexibilité et la généricité nécessaire pour s'adapter aux divers besoins des applications finales. Nous détaillons les principes et enjeux de chaque étape identifiée en présentant les approches existantes ainsi que les améliorations que nous leur apportons dans cette thèse. Enfin nous terminons par une étude des stratégies de distribution envisageables et leurs impacts sur les performances du système.

2.2.1 Formalisation de la chaîne de traitement

Notre système repose sur une formalisation de la chaîne de traitement en une série d'étapes fonctionnelles. Cette conceptualisation vise à intégrer et à factoriser les processus mis en jeu pour extraire à partir des images les différents types d'informations sur l'utilisateur. Voici les différentes étapes que nous avons identifiées :

Acquision : les images sont tout d'abord acquises de manière synchrone.

Traitements d'images : les images sont ensuite traitées localement pour extraire des primitives $2D$. En particulier les silhouettes de l'utilisateur sont calculées sous différents formats (image binaire et contours). Cette information sur la région d'intérêt est particulièrement intéressante pour les autres traitements. D'autres informations

telles que la position du visage ou des doigts peuvent aussi être extraites si une interaction les requiert.

Modélisation des formes : les silhouettes sont transmises aux processus en charge de la modélisation des formes pour obtenir un modèle géométrique 3D de la personne. Cette donnée relative au positionnement de l'utilisateur dans l'espace est requise par la plupart des interactions tri-dimensionnelles.

Modélisation du mouvement : le flux des modèles géométriques obtenus à chaque instant permet d'extraire par la suite le mouvement de la personne. Cette information comporte à la fois le suivi temporel de l'utilisateur et l'identification des parties rigides comme les membres du corps (tête, mains, pieds, ...).

Interaction et rendu : les informations de formes et de mouvement sont ensuite transmises aux applications interactives. Le modèle peut aussi être affiché (information photométrique) au sein de la scène virtuelle.

Notons que nous avons volontairement choisi d'extraire le mouvement à partir de la modélisation des formes et non le contraire comme le propose certains systèmes [Carranza 03]. En effet la modélisation des formes peut s'effectuer sans *a priori* sur la scène alors que la modélisation du mouvement nécessite certains *a priori* comme le modèle de l'utilisateur. Notre système permet ainsi de récupérer l'information de géométrie dans n'importe quelle situation avec des objets supplémentaires apportés par l'utilisateur par exemple.

Cette formalisation offre une grande généricité car elle intègre les grands types de données sur l'utilisateur : silhouettes, géométrie $3D$, suivi temporel, identification et photométrie. De plus une grande flexibilité est proposée lors de la mise en oeuvre du système car pour chaque étape plusieurs algorithmes peuvent être mis en jeu selon les besoins des applications finales. En particulier plusieurs modélisations de forme fonctionnant en parallèle peuvent être nécessaires. Par exemple pour effectuer à la fois un rendu visuel et des interactions volumiques, le système doit extraire par des méthodes différentes la surface de l'utilisateur et son occupation dans l'espace.

2.2.2 Les différentes étapes

Pour comprendre l'ensemble du système, nous décrivons brièvement chacune des étapes de notre chaîne de traitement en montrant leurs enjeux. La figure 2.4 représente de façon schématique l'ensemble de ces étapes. Les détails des algorithmes et des implantations sont donnés dans les chapitres suivants.

Acquisition synchrone des images

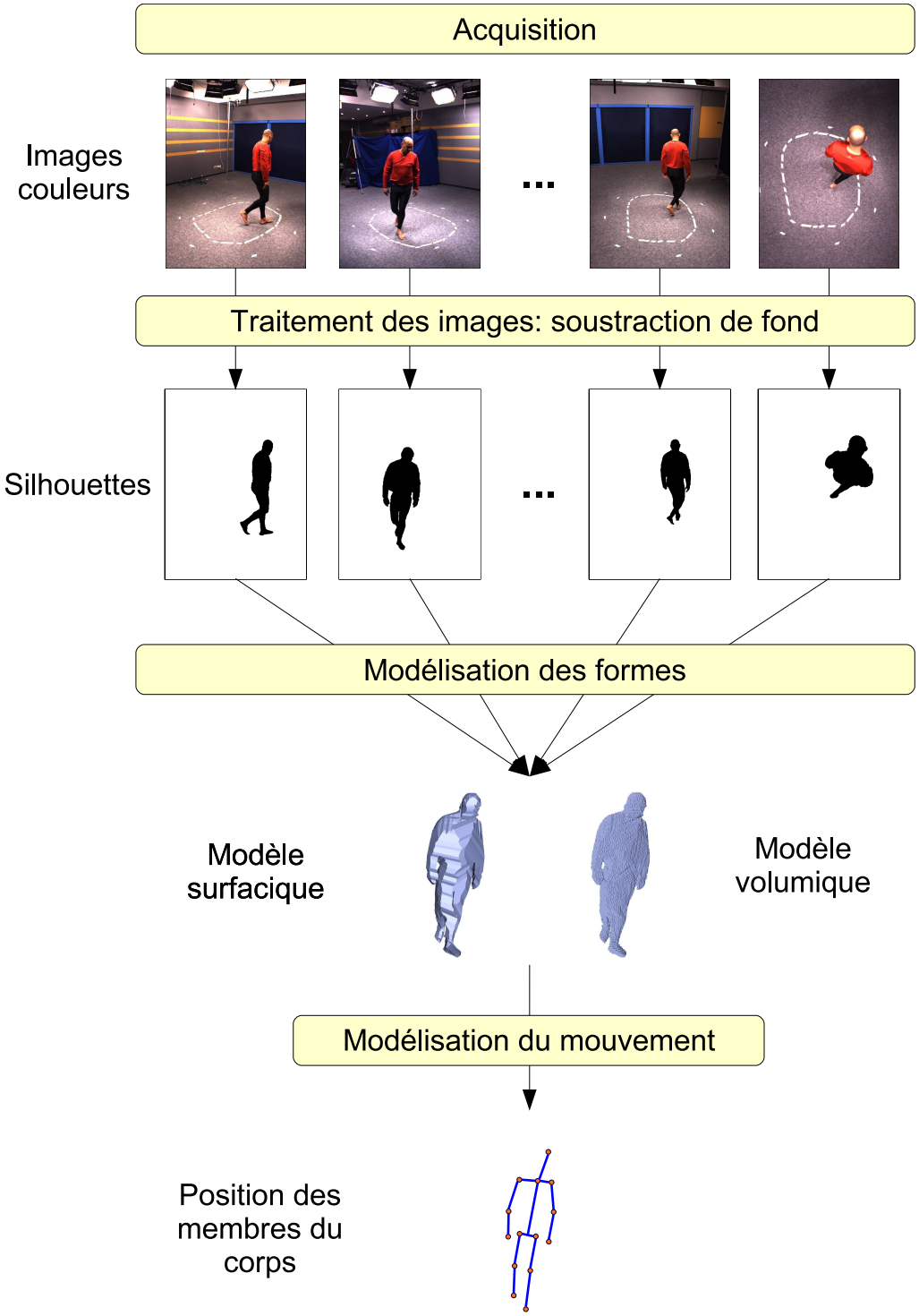


Figure 2.4 *Chaîne complète de traitement avec les différentes étapes : acquisition des images, traitements des images, modélisation des formes, modélisation du mouvement.*

L'acquisition des images filmées par les caméras nécessitant une importante bande passante, un ensemble de PCs est dédié à cette tâche. Une synchronisation des capteurs entre eux est nécessaire pour obtenir au sein d'une trame des images correspondant au même événement dans la scène. Dans la plupart des cas des systèmes matériels comme des boîtiers électroniques de synchronisation sont utilisés. Chaque image est alors estampillée du numéro de trame à laquelle elle correspond.

Extraction de la silhouette

Les régions d'intérêt des images sont ensuite identifiées pour ne retenir que la partie correspondant à l'utilisateur : la *silhouette*. Cette étape est habituellement appelée *soustraction de fond*. Une solution très utilisée dans l'audiovisuel, dénommée *chroma-keying*, repose sur l'utilisation d'un fond de couleur uniforme et en la détection de cette couleur dans les images. Des solutions moins contraignantes ont aussi été proposée par la communauté de la vision par ordinateur. Elles reposent sur l'apprentissage d'un modèle du fond supposé fixe et sur la comparaison de chaque pixel à ce modèle [Horprasert 99, Cheung 00, Shen 04]. La difficulté est alors de réaliser cette tâche de manière suffisamment robuste tout en respectant les contraintes de performance. Des filtres morphologiques (érosion/dilatation) sont souvent utilisés pour supprimer le bruit des silhouettes extraites.

Vectorisation de la silhouette

Les silhouettes obtenues par la soustraction de fond sont représentées sous la forme d'images binaires : une valeur 0 ou 1 est associée à chaque pixel. Or certains algorithmes de modélisation des formes utilisent plutôt les silhouettes en tant que polygones dans l'image. Ces deux représentations de la silhouette sont duales et le traitement pour passer de la forme image à la forme polygonale est appelée *vectorisation*. Certains comme Matusik *et al.* [Matusik 01] utilisent un algorithme de type Marching Cubes pour effectuer cette vectorisation. Ces approches ne sont cependant pas parfaitement adaptées aux problèmes car elles génèrent de très nombreux petits segments. D'autres algorithmes [DebledRennesson 95, DebledRennesson 04] ont été proposés permettant d'obtenir des segments plus longs tout en conservant des garanties sur la précision de cette vectorisation.

Modélisation des formes

Pour permettre une interaction avec un univers virtuel tri-dimensionnel, il est nécessaire d'obtenir des informations $3D$ sur la position de l'utilisateur à partir des informations $2D$ précédemment extraites. Plus précisément il est nécessaire de calculer un modèle complet de la géométrie de l'utilisateur. Cette modélisation des formes a été largement étudiée par la communauté de vision par ordinateur. La plupart de ces travaux visent cependant à obtenir les modèles les plus précis possibles sans se préoccuper des coûts de calculs, ceux-ci étant dans la plupart des cas effectués hors-ligne. Or l'interacti-

vité du système impose une exécution temps réel. Il est donc nécessaire de développer des techniques spécifiques cherchant le meilleur compromis entre précision des modèles et rapidité des calculs. Dans cette thèse nous proposons à cet effet différentes modélisations des formes adaptées aux systèmes d'interaction. Nous nous sommes particulièrement intéressés à la parallélisation de ces algorithmes pour améliorer la latence. Une étude précise de ces méthodes distribuées est réalisée dans la partie II.

Modélisation du mouvement

La modélisation des formes fournit la géométrie instantanée de l'utilisateur. Cette information est suffisante pour beaucoup d'applications mais certaines applications requièrent des informations supplémentaires. Tout d'abord l'évolution de ces géométries dans le temps, c'est-à-dire le **dynamisme** de la scène, est très important pour permettre un suivi de certains points particuliers de l'utilisateur. De plus l'**identification** des parties de l'utilisateur (main, pied, ...) apporte une sémantique à ces formes et permet d'appliquer des traitements spécifiques à ces régions.

Plusieurs approches existent déjà pour extraire ces informations. En ce qui concerne le dynamisme, certaines méthodes [Matsuyama 04] reposent sur une déformation du maillage calculée à la trame t pour passer au modèle de la trame suivante $t + 1$. Cette déformation leur permet alors de mettre en correspondance les modèles de trames consécutives. Cependant les déformations ne correspondent pas toujours aux mouvements réels en particulier lors de déplacements importants. Les techniques de déformation de maillage sont aussi très complexes à mettre en oeuvre à cause des problèmes d'auto-intersection et de changement de topologie. Elles s'avèrent donc coûteuses à l'exécution et mal-adaptées pour les besoins des systèmes d'interaction.

D'autres approches apportent simultanément une réponse aux deux problèmes – le dynamisme et l'identification – via un suivi de mouvement humain (*Human Motion Tracking*). Ces approches utilisent l'hypothèse que nous filmons un être humain : elles consistent à détecter et suivre les différentes parties de l'utilisateur (bras, tête, jambes, ...). De très nombreux travaux ont été récemment publiés dans la communauté de la vision par ordinateur pour répondre à ce problème. Un état de l'art de ces techniques est présenté chapitre 8 (page 89). Ces méthodes se révèlent cependant mal adaptées aux systèmes interactifs. Leurs temps de calcul sont souvent beaucoup trop élevés pour envisager aujourd'hui une exécution temps réel. En outre la plupart d'entre-elles requièrent la connaissance des mensurations de l'utilisateur, limitant l'accès au système d'interaction à un nombre restreint de personnes connues.

Nous proposons dans cette thèse, dans la partie IV page 87, une méthode nouvelle de modélisation de mouvement adaptée aux systèmes d'interaction. Cette méthode permet de retrouver l'évolution de la position des membres de l'utilisateur en nécessitant le minimum d'informations sur la forme géométrique de celui-ci. Cette méthode présente

l'avantage d'être indépendante de l'épaisseur des membres.

Interactions et rendu

Les informations de forme et de mouvement sont ensuite transmises aux applications finales. Chaque méthode d'interaction ayant ses propres besoins, différents algorithmes de modélisation peuvent être nécessaires en parallèle pour construire l'application globale. Par exemple les collisions avec des objets rigides utilisent des modèles de surface triangulée alors que celles avec des objets volumiques (fluides, etc...) utilisent plutôt des représentations volumiques des formes. Ces besoins spécifiques s'expriment aussi dans la qualité des interactions souhaitées. Si nous souhaitons faire des collisions simplement statiques avec des objets, la modélisation des formes est alors suffisante. Cependant si nous souhaitons obtenir des collisions dynamiques, comme donner de l'inertie à l'objet quand on le touche, alors il est nécessaire d'obtenir la vitesse de l'utilisateur au point de contact et donc d'avoir des informations sur son mouvement. Plus les interactions souhaitées deviennent complexes, plus le besoin en information spécifique sur l'utilisateur devient important. Ceci implique de mettre en jeu plus de traitements en parallèle pour obtenir l'ensemble de ces données en temps réel.

Lors de la visualisation de la scène virtuelle, il est parfois nécessaire d'afficher l'utilisateur immergé dans cet univers. Pour ce faire il est nécessaire d'effectuer un rendu du modèle de l'utilisateur le plus fidèlement possible. Pour cela nous utilisons un modèle de forme précis surfacique. Les images couleurs issues des différentes caméras sont associées à ce modèle géométrique pour effectuer le rendu à travers un nouveau point de vue. La technique consiste à "plaquer" les images des différentes caméras comme des textures sur le modèle en suivant une approche similaire à celles proposées par Debevec *et al.* [Debevec 96] et par Li *et al.* [Li 04]. L'affichage de l'utilisateur de façon cohérente dans l'univers virtuel nécessite un travail supplémentaire pour obtenir un bon réalisme : rééclairage pour que l'illumination soit cohérente, calcul des ombres projetées au sol ou de l'auto-ombrage.

2.2.3 Stratégies de distribution

La conception d'un système multi-caméra passe nécessairement par une gestion de la distribution des tâches sur l'ensemble des ressources disponibles. Tout d'abord, les limitations sur les débits des entrées/sorties des machines actuelles limitent le nombre de caméras gérées par une seule machine à 2 ou 3. Pour utiliser plus de caméras, il est nécessaire d'utiliser plusieurs ordinateurs. De plus, les contraintes de performance temps réel impliquent de tirer au mieux parti de l'ensemble des ressources de calcul disponibles. La distribution des calculs sur un groupe de processeurs apparaît donc incontournable. Cela pose cependant de nombreuses difficultés. L'extraction d'informations

tri-dimensionnelles à partir de plusieurs caméras diffère en effet des problèmes généralement traités par la communauté du parallélisme selon plusieurs aspects :

- **la nature du problème :** le parallélisme s'intéresse principalement à des problèmes très coûteux comme le calcul scientifique ou à des problèmes manipulant des données excessivement volumineuses comme les tris dans des grandes bases de données. Ces algorithmes sont *unitaires*, c'est-à-dire qu'ils ne sont pas composés d'une chaîne de traitement. Ces problèmes sont relativement *homogènes* comparés à la grande diversité des tâches qui cohabitent dans un système multi-caméra : traitement d'images, calcul géométrique 2D / 3D. Une unique stratégie de parallélisation, comme un découpage de l'espace en zones, ne peut donc pas être appliqué à l'ensemble du système. Chaque traitement ayant ses propres spécificités, une parallélisation spécifique à chacun doit être développée.
- **les critères de performance :** dans le cadre du parallèlisme, le but est de minimiser le temps de traitement pour une certaine quantité de données fixée. Dans notre cas, les données sont un flux permanent de trames et cette mesure n'est donc pas bien adaptée. Comme nous l'avons présenté en introduction les performances du système s'expriment par le débit (lié au temps pour traiter un nombre fixé de trames) et surtout la latence (liée au temps pour traiter une trame). Il est donc nécessaire de réétudier les stratégies de distribution dans ce contexte multi-critère, et ainsi d'analyser leur impact sur ces critères.
- **la granularité** (taille des tâches à paralléliser) : les temps d'exécution des programmes étudiés en parallélisme relèvent de l'heure ou de la minute. Dans notre cas le traitement d'une trame doit être réalisée en un dizième de seconde. Ceci rend très complexe la distribution des calculs car les échanges de données entre processeurs prennent un temps considérable par rapport aux temps de calcul.

Le problème de la distribution d'un système multi-caméra est donc très particulier. La plupart des systèmes proposés dans la littérature comportent une part de distribution des tâches. Cheung *et al.* [Cheung 00] ou Gross *et al.* [Gross 03] proposent d'effectuer les tâches de traitement d'image localement sur chaque machine dédiée à une caméra. La modélisation $3D$ reste séquentielle et exécutée sur une machine centrale. Davis *et al.* [Borovikov 03], Matsuyama *et al.* [Wu 06] ou le projet Tele-Immersion [Kelshikar 03] vont plus loin en distribuant les calculs internes de la modélisation sur plusieurs processeurs. D'autres approches utilisent indirectement la parallélisation en utilisant le GPU comme Hasenfratz *et al.* [Hasenfratz 04]. Cependant ces travaux se limitent à proposer des solutions ad-hoc sans véritablement identifier les stratégies de distribution mises en jeu. Dans le cadre de cette thèse nous proposons un tel travail d'identification. Ceci nous permet de mieux comprendre les pré-requis de chaque technique de parallélisation et l'impact qu'elles ont sur les performances du système en terme de débit et de latence. Nous distinguons deux grandes catégories de stratégies de parallélisation selon qu'elles s'intéressent au flux des trames (*parallélisation par flux*) ou aux calculs internes des traitements

liés à une seule trame (*parallélisation par trame*).

Pour cette étude, nous introduisons tout d'abord quelques définitions et notations utiles que nous utiliserons dans la suite de ce document. Dans le contexte du parallélisme, nous désignerons par le mot *nœud* un ordinateur/processeur impliqué dans le traitement distribué. Nous noterons la latence L et le débit D. Pour simplifier certaines formules nous exprimerons le débit par le délai entre deux trames consécutives $d = \frac{1}{D}$.

2.2.3.1 Parallélisation par flux

Un système multi-caméra traite une séquence de trames, c'est à dire un *flux* multi-image. Une idée classique pour accélérer des applications traitant des flux est d'utiliser un *pipeline*. Cette stratégie a été particulièrement étudiée dans le domaine de l'architecture des processeurs. L'application est alors scindée en une séquence d'étapes notées E_i (voir figure 2.5(a)). La responsabilité du traitement de chaque étape est alors attribuée à différents processeurs dédiés. Ceci permet à un nœud associé à une étape de l'algorithme de traiter la trame t, pendant que le nœud chargé de l'étape suivante du pipeline traite la trame $t - 1$. Le pipeline permet d'augmenter le débit en autorisant à plusieurs trames d'être traitées simultanément (voir figure 2.5(b)). Les différentes étapes sont en général naturellement déduites de la structure du système à paralléliser. Le choix de ces étapes est très important pour les performances. Un découpage en trop peu d'étapes ne permettra d'exploiter au maximum le potentiel de distribution de l'application. En revanche un découpage en un trop grand nombre d'étapes sera désavantageux, les communications entre ces étapes risquant de devenir un facteur limitant. En utilisant une seule machine, le débit est fonction de la somme des temps de traitement d'une trame par l'étape E_i, noté t_i. Ainsi $d = \sum t_i$ (voir figure 2.5(a)). Lorsque le système utilise un pipeline distribué, le débit est fonction du temps d'exécution de l'étape la plus lente : $d = \max t_i$ (voir figure 2.5(b)). Dans le cas idéal où les étapes ont des coûts identiques, le débit est alors multiplié par un facteur égal à la longueur du pipeline.

Dans le cas où les temps des étapes sont irréguliers (voir figure 2.5(b)) le débit est limité par la tâche la plus lente. Les processeurs dédiés aux autres tâches plus rapides ne sont alors pas exploités totalement. Pour palier à cela et rééquilibrer la charge du pipeline nous pouvons tirer partie de certaines propriétés des tâches. Une étape est dite *statiquement calculable* si elle n'utilise pas de cohérence temporelle, c'est à dire qu'elle traite la trame t indépendamment des informations liées aux trames précédant t. Cette propriété permet d'utiliser plusieurs nœuds pour le traitement de l'étape (voir figure 2.5(c)). On parle alors d'un ensemble d'*unités de calcul* d'une même étape : une nouvelle trame t peut être traitée dès qu'une des unités de calcul de l'étape est disponible. Le nombre d'unités de calcul, noté n_i, doit être suffisemment grand pour éviter toute attente de traitement d'une trame. Il doit donc être ajusté selon la durée du traitement et sa proportion par rapport aux autres étapes. Le débit devient ainsi : $d = \max \frac{t_i}{n_i}$. Adapter cette tech-

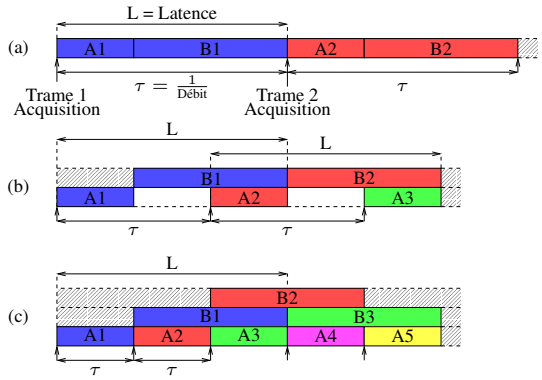


Figure 2.5 *Parallélisation par flux. Supposons que A et B sont les deux étapes de calcul de notre programme. At et Bt correspondent au traitement de la trame t. Chaque ligne correspond à un processeur. Les blocs colorés correspondent à l'exécution d'une tâche et les blocs blancs à des périodes d'inactivité. Le schéma (a) représente une exécution purement séquentielle, le schéma (b) une exécution avec un pipeline à 2 étapes. Le schéma (c) ajoute une seconde unité de calcul pour la seconde étape B du pipeline.*

nique à une étape non statiquement calculable peut s'avérer réalisable mais nécessite des techniques d'ordonnancement avancées et des communications supplémentaires non détaillées ici. Notons que lorsque le système n'est composé que d'étapes statiquement calculables, celles-ci peuvent être réunies en une seule tâche. La parallélisation par flux revient alors à avoir p serveurs – un par processeur – chacun pouvant effectuer cette tâche : le débit est alors parfaitement multiplié par p.

2.2.3.2 Parallélisation par trame

Les techniques de distribution précédentes peuvent améliorer significativement le débit, en permettant d'augmenter le nombre de trames traitées simultanément. Cependant, elles introduisent une latence supplémentaire du fait des communications réseaux introduites entre les nœuds. Pour diminuer la latence et donc améliorer la réactivité du système, il nous faut réduire le temps global de traitement d'une trame. Ceci nécessite de distribuer le travail effectué au sein même d'une trame sur p_i processeurs. Nous introduisons donc un niveau de granularité supplémentaire dans l'architecture : non seulement le pipeline se subdivise en unités de calcul qui sont affectés à un étage du traitement pour une seule trame donnée, mais l'unité elle-même peut comprendre plusieurs nœuds, pour le traitement parallèle d'une étape à une trame donnée (voir figure 2.6).

En considérant l'ensemble des parallélisations nous obtenons :

$$L = \sum t_i^{//} \geq \sum \frac{t_i}{p_i}$$

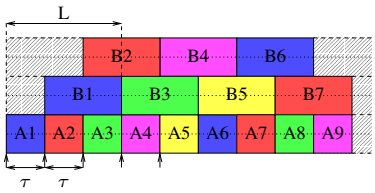


Figure 2.6 *Parallélisation par trame (deux processeurs pour l'étape A et quatre processeurs pour l'étape B) : amélioration de la latence par rapport à la parallélisation par flux (voir figure 2.5).*

$$d = \max \frac{t_i^{//}}{n_i} \geq \max \frac{t_i}{p_i \times n_i}$$

Les inégalités représentent les coûts liés à la parallélisation, en particulier les coûts de communication et le déséquilibre de la charge de travail entre processeurs. Le cas d'égalité est atteint lorsque la parallélisation est idéalement efficace.

2.3 Bilan

Un système d'interaction permet d'acquérir différents types d'informations sur l'utilisateur allant du suivi de points à l'identification des données en passant par des informations globales telles que la géométrie complète de l'utilisateur.

Nous avons présenté les différentes solutions existantes en les regroupant en grandes familles d'approches selon les informations extraites. Les systèmes à base de marqueurs fournissent des données très précises mais sont contraignants. Des systèmes utilisant des caméras vidéo ont démontré leur potentialité pour extraire des informations plus globales et sans nécessiter d'équipement porté par l'utilisateur. Toutefois, pour obtenir des informations $3D$, il est nécessaire de considérer plusieurs caméras filmant la scène depuis différents points de vue. C'est dans ce contexte des systèmes multi-caméras que nous avons placé mes travaux de thèse.

Ces systèmes multi-caméras sont plus complexes que la simple juxtaposition de blocs mono-caméra et nécessitent la fusion des données en provenance des caméras pour obtenir les informations souhaitées. L'état de l'art réalisé fait apparaître que la plupart des approches proposées se concentrent sur un seul type d'information. Elles ne proposent pas de solutions génériques pour extraire l'ensemble des types d'information. De plus on constate l'absence de systèmes multi-caméras offrant simultanément des données de bonne précision et des performances temps réel.

Le premier problème concerne la généricité et la flexibilité du système. Pour cela nous avons proposé un formalisme de la chaîne de traitement en nous basant sur une synthèse des approches existantes. Les informations images $2D$, en particulier les silhouettes, sont d'abord extraites. Ces dernières sont transmises à l'étape de modélisation des formes permettant d'obtenir une information sur la géométrie de la scène. Remarquons que cette information peut être calculée sous plusieurs formes comme nous le verrons dans la partie II. Cette information de forme est ensuite utilisée par la modélisation du mouvement (partie IV) qui permet d'obtenir simultanément les informations de suivi et d'identification. Enfin l'ensemble de ces données sur l'utilisateur est transmis aux applications finales. Nous avons fourni dans ce chapitre un aperçu de ces étapes : les parties suivantes de la thèse s'attacheront à détailler chacune d'entre-elles.

Le second problème concerne le coût des calculs nécessaires pour obtenir une bonne précision et gérer un nombre important de caméras. Pour répondre à ce problème, la parallélisation des calculs sur l'ensemble des ressources disponibles apparaît comme naturelle. L'utilisation de plusieurs caméras implique en effet la présence de plusieurs machines. Le parallèlisme apparaît d'autant plus incontournable qu'aujourd'hui l'augmentation de la puissance des ordinateurs passe par la multiplication des cœurs de calcul. Nous avons fourni une étude des différentes stratégies qui s'offrent à l'utilisateur : deux grands types d'approches apparaissent. La parallélisation par flux permet d'utiliser la notion de pipeline en distribuant les étapes sur des processeurs distincts. Cette stratégie permet d'améliorer le débit mais a un impact négatif sur la latence. Pour améliorer cette dernière, il est nécessaire d'effectuer une parallélisation par trame, c'est-à-dire de distribuer les calculs internes d'une étape. Si la parallélisation par flux est générique, la parallélisation par trame est spécifique à chaque algorithme et est donc plus difficile. Les parallélisations que nous présentons dans cette thèse concernent ainsi principalement la distribution des calculs internes pour chaque algorithme.

II

Modélisation des formes

État de l'art 3

La problématique de la modélisation des formes consiste à calculer la géométrie tridimensionnelle d'un objet à partir d'images fournies par plusieurs caméras selon différents points de vue. Nous supposerons dans la suite que ces caméras sont préalablement calibrées et que les images sont synchrones. Différentes grandes familles de modélisation des formes ont été proposées dans la littérature. Nous présentons dans un premier temps (section 3.1) ces familles en montrant leurs avantages et inconvénients. Cette analyse montre que les approches utilisant le modèle d'*enveloppe visuelle* apportent aujourd'hui un des meilleurs compromis entre qualité et performance. Ces approches semblent donc bien adaptées aux systèmes d'interaction. Nous nous intéressons ensuite (section 3.2) plus spécifiquement à cette catégorie d'approches et détaillons les diverses méthodes qui ont été proposées par la communauté de la vision par ordinateur.

3.1 Catégories de modélisation des formes

3.1.1 Les méthodes actives

Les méthodes actives reposent sur la projection de motifs lumineux connus sur la scène. Ceux-ci permettent en les détectant dans les images de retrouver facilement la structure $3D$ de la scène.

Les scanners lasers projettent des primitives telles que des points ou des lignes dans des directions connues. Ces primitives sont facilement identifiées dans les images grâce à la forte luminosité des zones ainsi éclairées. Les positions des points éclairés

(a) Scanner laser (b) Lumière structurée (c) Caméra temps de vol

Figure 3.1 *Différents système de modélisation active projetant des informations sur la scène.*

peuvent ainsi être très précisément calculées. Cette opération est répétée dans différentes directions pour couvrir l'ensemble de la scène. Cette technique fournit alors un nuage de points $3D$ extrêmement précis. Ces méthodes ont connus un grand succès pour la modélisation d'oeuvre archéologique comme des statues [Levoy 00] (voir figure 3.1(a)). Cependant, l'acquisition est assez lente (plusieurs secondes au minimum) et ne peut donc pas s'appliquer à des objets dynamiques.

D'autres approches, appelées modélisations par lumière structurée, utilisent quant à elles des motifs $2D$ souvent projetés à l'aide de vidéo-projecteurs [Rusinkiewicz 02]. Ces motifs sont généralement des lignes verticales ou horizontales (voir figure 3.1(b)). Connaissant la calibration (position et orientation) du vidéo-projecteur par rapport à la caméra, la profondeur de chaque pixel peut être calculée en cherchant la position du motif lui correspondant. On obtient alors la carte de profondeur, c'est-à-dire la distance entre la caméra et l'objet pour chaque pixel. Cette information ne fournit pas un modèle complet de la scène : si la caméra filme l'avant d'un utilisateur, aucune donnée n'est alors disponible sur son dos. Pour obtenir un modèle géométrique $3D$ complet de la scène, il est nécessaire d'obtenir plusieurs cartes de profondeur depuis plusieurs points de vue [Waschbüsch 05]. Or ces systèmes sont très vite limités (2 ou 3) en le nombre de points de vue à cause des interférences entre les différentes figures projetées. De plus les motifs projetés perturbent l'extraction d'informations photométriques. Certains travaux apportent des solutions à ce problème en rendant le motif invisible pour une caméra standard [Cotting 04]. Le vidéo-projecteur se comporte alors comme un projecteur de lumière blanche éblouissant l'utilisateur lorsqu'il est face à celui-ci.

Une autre technologie active consiste à utiliser la vitesse de propagation de la lumière pour mesurer la distance entre le capteur et l'utilisateur [Gokturk 04]. Pour cela un flash dans l'infra-rouge est émis ; le capteur mesure alors le temps séparant l'émission du flash

et la réponse due à sa réflection sur l'utilisateur. Ce temps correspond à l'aller-retour de la lumière. Bien que très prometteuse, ces caméras temps de vol (*Time Of Flight*) sont pour l'instant limitées à de faibles résolutions (128×128) et à des distances pas trop élevées (maximum 2 m). De plus la précision des mesures décroît fortement lorsque la scène est illuminée par la lumière naturelle ; la lumière infra-rouge du soleil en particulier perturbe la détection de la réflection du flash.

Les méthodes actives permettent d'obtenir des modèles tri-dimensionnels sur la scène de manière relativement précise. Les motifs projetés perturbent cependant cette scène ou gênent l'utilisateur. Ces techniques sont donc difficilement utilisables dans un système d'interaction.

3.1.2 Les méthodes passives

Pour éviter de perturber la scène en y ajoutant des informations, des travaux proposent des méthodes se basant uniquement sur les informations naturellement disponibles. Il existe de nombreuses familles d'approches dites passives : celles-ci peuvent être organisées en fonction des informations $2D$ utilisées.

Certaines méthodes utilisent les silhouettes de l'objet à modéliser. En recoupant ces informations issues des différents points de vue, la géométrie de cet objet est calculée. Le modèle le plus généralement utilisé est l'enveloppe visuelle, défini par Baumgart [Baumgart 74] et baptisé par Laurentini [Laurentini 94]. Il correspond à l'enveloppe maximale dont la projection dans les différents points de vue est intérieur aux silhouettes. Un point particulièrement intéressant dans l'enveloppe visuelle est le fait que l'objet réel est forcément inclus dans celle-ci. Le calcul de ces modèles est relativement stable et robuste aux bruits d'entrée. Les approches existantes permettent d'obtenir un modèle assez précis avec une dizaine de caméras en moins d'une seconde. Ce modèle n'est cependant qu'une approximation de la scène et ne contient en particulier pas les concavités. D'autres modélisations comme [Boyer 97] proposent d'intégrer d'avantage d'information comme les contours internes liés à des bords de l'objet. L'obtention de tels contours reste cependant relativement difficile et nécessite souvent des techniques actives [Crispell 06].

D'autres approches utilisent l'information photométrique pour mettre en correspondence des zones de différents points de vue. Ces appariements permettent ensuite par triangulation d'obtenir des coordonnées $3D$ sur la surface de la scène à modéliser. Les premières approches photométriques ont concerné la stéréo binoculaire : la scène est filmée depuis deux points de vue relativement proches imitant ainsi le système de vision humaine. Le résultat obtenu est alors généralement une carte de profondeur. De très nombreuses techniques ont été proposées et nous ne pouvons pas les détailler ici. Pour plus de détails sur ces techniques et leur comparaison nous renvoyons le lecteur intéressé à l'état de l'art réalisé par Scharstein et Szeliski [Scharstein 02]. Ces techniques ont aussi fait l'objet d'implantations parallèles [Fua 91] et plus récemment sur GPU [Gong 05]

leur permettant de s'exécuter en temps réel. Si ces techniques fonctionnent bien dans les cas d'école, leur utilisation dans la pratique est très délicate : les images sont souvent trop bruitées et la scène pas assez texturée. Les modèles alors obtenus contiennent de très nombreuses erreurs ou incertitudes. Plus récemment, la communauté de vision s'est penchée sur le problème de la modélisation photométrique à partir de plusieurs points de vue éloignés, dénomé stéréo multi-vue. Une des difficultés de ces approches concerne la gestion de la visibilité dans chaque image. Une autre difficulté provient du fait qu'un point de la surface n'a pas exactement le même aspect selon le point de vue (surface non lambertienne ou caméras ayant des réglages différents). Seitz *et al.* [Seitz 06] présentent un état de l'art très récent de ces techniques. Malgré des implantations récentes sur GPU [Labatut 06], ces méthodes sont très coûteuses et nécessitent plusieurs secondes voire plusieurs minutes pour modéliser une trame avec une dizaine de caméras. De plus elles nécessitent dans la plupart des cas l'intégration des contraintes liées aux silhouettes pour la stabilité des calculs. Le gain en qualité dans le cadre de la modélisation d'un être humain n'est pas forcément significatif, ce dernier comportant peu de concavités.

Mentionnons enfin l'existence d'approches cherchant à modéliser la forme de l'utilisateur en fonction de l'éclairage/ombrage (*Shape from Shading* [Zhang 99]). Ces approches restent cependant très délicates et ne fournissent pas une information fiable et complète dans des situations génériques.

3.2 Méthodes de modélisation d'enveloppes visuelles

L'enveloppe visuelle s'avère être ainsi le modèle offrant un des meilleurs compromis entre précision du modèle obtenu et rapidité des calculs. Les approches stéréo multi-vue, bien que très prometteuses, restent beaucoup trop lentes et n'apportent pas obligatoirement un gain en précision très significatif. Dans le contexte des systèmes d'interaction, nous choisissons donc de nous focaliser sur les approches de calcul d'enveloppes visuelles. Plusieurs types d'approches existent :

- Les approches **volumiques** se basent sur une représentation discrète de l'espace et cherchent l'occupation de chaque zone de l'espace.
- Les approches **surfaciques** cherchent à obtenir géométriquement la surface de l'enveloppe visuelle en se basant principalement sur les contours et non l'intérieur des silhouettes.
- Les approches **hybrides** permettent de conjuguer ces deux approches en fournissant à la fois une surface et une représentation volumique.

Nous devons aussi mentionner l'existence d'approches images permettant de générer l'enveloppe visuelle sous un nouveau point de vue [Matusik 00] (*Image Based Rendering*). Comme ces approches ne permettent pas de récupérer réellement des informations mé-

triques sur la forme tridimensionnelle de la scène, nous ne les considérons donc pas.

3.2.1 Les approches volumiques

Les approches volumiques se basent sur une discrétisation de l'espace en cellules élémentaires, les *voxels*. L'occupation de chaque voxel est décidée de façon très simple : un voxel appartient à l'enveloppe visuelle si et seulement si sa projection dans chaque image est incluse dans la silhouette. L'espace est généralement discrétisé à l'aide d'une grille régulière de voxels parallélépipédiques. Proposée à l'origine par Martin et Aggarwal [Martin 83], de nombreuses améliorations ont été apportées par Niem [Niem 94] et par Cheung [Cheung 00]. Des implantations parallèles ont aussi été proposées par Borovikov *et al.* [Borovikov 00] et Wada *et al.* [Wada 00] ainsi que des implantations GPU par Hasenfratz *et al.* [Hasenfratz 03]. Pour concentrer les calculs sur les zones d'intérêts et pour ainsi accélérer la modélisation, certaines approches [Chien 86, Szeliski 93] utilisent des discrétisations hiérarchiques sous la forme d'octrees. Des travaux se sont aussi intéressés sur la prise en compte d'autres critères comme la régularité de la surface en utilisant des graph-cut [Boykov 03]. Ces méthodes ayant connu un très fort succès dans les années 90, de très nombreuses améliorations ont été apportées et nous ne pouvons les décrire de manière exhaustive. Dyer [Dyer 01] et Slabaugh *et al.* [Slabaugh 01] proposent des états de l'art de ces méthodes de modélisation volumique d'enveloppes visuelles. Le succès de ces approches reposent principalement dans la simplicité de leur implantation et dans leur complexité linéaire en le nombre de caméras. Cependant ces approches présentent un rapport désavantageux entre le coût des calculs et la précision. Une précision arbitraire peut être atteinte, mais au prix d'une augmentation de la résolution de la discrétisation. D'autre part les modèles obtenus sont souvement mal adaptés pour la visualisation, la surface apparaissant "crênelée" (voir figure 3.2(a)) ou bien nécessitant l'utilisation d'une technique de *marching-cube* et générant alors un nombre très grand de petites facettes.

3.2.2 Surfaciques

Les approches surfaciques cherchent à retrouver directement la surface de l'enveloppe visuelle. Ces approches reposent sur le fait que pour chaque point situé sur un contour d'une silhouette, la ligne de vue associée est tangente à la surface de l'objet. Ainsi en fusionnant ces informations de tangence, un modèle géométrique $3D$ de la scène peut être obtenu. Plus concrètement l'enveloppe visuelle correspond à l'intersection des cônes de vue. Le sommet de ces cônes est le centre optique et leurs supports correspondent aux contours des silhouettes.

[Koenderink 84, Cipolla 92, Boyer 97] reconstruisent des points isolés en utilisant des approximations locales du second ordre de la surface. L'obtention d'une surface fermée à partir de ces nuages de points reste très délicate. En particulier celle-ci est difficile aux voisinages des points frontières.

(a) Modélisation volumique (b) Modélisation surfacique (c) Modélisation hybride

Figure 3.2 *Trois catégories d'approches de calcul d'enveloppe visuelle avec* 6 *caméras. (a) Modélisation volumique avec une grille régulière de voxels parallélipipédiques* $64 \times 64 \times 64$. *(b) Modélisation surfacique avec l'algorithme proposé par Jean-Sébastien Franco. (c) Modélisation hybride*

D'autres approches utilisent le fait que l'enveloppe visuelle est un polyèdre lorsque les contours sont représentés de façon polygonalisée.

Une première approche de calcul de l'enveloppe visuelle polyèdrique a été proposée par Baumgart *et al.* [Baumgart 74] dès 1974. Plus tard des travaux comme celui de Hughes *et al.* [Laidlaw 86] ont proposé des approches en lien avec la géométrie algorithmique. Elles utilisent des techniques dites de géométrie constructive des solides, CSG. Leur principe repose sur une représentation des volumes sous la forme d'unions ou d'intersections entre des solides de base tels que des cubes ou des prismes. Chaque cône de vue est décomposé en solides de base : on calcule ensuite leur intersection. On obtient alors l'enveloppe visuelle sous une forme de compositions de volumes de base. La surface est calculée en utilisant des techniques complexes développées par la communauté des CSG. Ces approches améliorent la méthode proposée par Baumgart mais ne traitent pas le problème des cas dégénérés. Elles sont aussi très coûteuses en calculs puisqu'elles nécessitent de très nombreux calculs $3D$.

Plus récemment, Matusik *et al.* [Matusik 01] proposent une solution géométrique n'effectuant que des calculs $2D$. Cette technique repose sur la nature projective de l'enveloppe visuelle mise en avant par Lazebnik *et al.* [Lazebnik 01]. Cela permet de calculer une bande de l'enveloppe visuelle pour chaque image. Le résultat est alors un ensemble de bandes non connectées entre-elles. Une solution pour les reconnecter, proposée par Ma-

tusik *et al.* , consiste à utiliser des règles de proximité pour identifier les points communs entre bandes. Malheureusement une telle heuristique ne permet par de garantir l'obtention d'un modèle fermé sans auto-intersections. Pour remédier à cela Franco *et al.* [Franco 03] présentent un algorithme d'exploration des arêtes et sommets de l'enveloppe visuelle permettant d'obtenir un modèle correspondant exactement à l'enveloppe visuelle polyèdrique (voir figure 3.2(b)).

3.2.3 Hybrides

Les méthodes volumiques et surfaciques comportent chacune leurs avantages et inconvénients. Les méthodes volumiques permettent d'obtenir facilement l'occupation d'un point donné de l'espace. Les méthodes surfaciques permettent quant à elles d'obtenir une surface de grande précision comparée aux méthodes volumiques. Il est à noter qu'il est très difficile de passer d'une représentation à l'autre. Pour bénéficier des avantages des deux méthodes simultanément, Boyer *et al.* [Boyer 03] proposent une solution hybride bénéficiant des possibilités des deux méthodes (voir figure 3.2(c)). Comme pour les méthodes surfaciques, cette technique extrait préalablement un nuage de points à la surface de l'enveloppe visuelle. Ce nuage de points permet de calculer un découpage de l'espace adapté à la surface, c'est-à-dire dont les facettes des cellules reposent sur les points de cette surface. Pour cela une tétraédrisation de Delaunay est calculée à partir du nuage de points. L'espace est ainsi découpé en cellules tétraédriques dont chaque sommet est un point de la surface de l'enveloppe visuelle. Comme pour les méthodes volumiques, les cellules internes à l'enveloppe visuelle sont alors identifiées. Cette représentation permet d'obtenir à la fois une forme volumique (tétraédres internes) et une forme surfacique en identifiant les facettes triangulaires à la frontière de l'enveloppe visuelle. Cette surface ne comporte pas le problème d'aliasing lié aux grilles parallélépipédiques. Le nombre de facettes généré est aussi beaucoup plus faible qu'en utilisant un algorithme de marching-cube sur une grille régulière de voxels.

3.3 Bilan

Ce chapitre a présenté les différentes techniques de modélisation des formes proposées principalement par la communauté de la vision par ordinateur. Malgré des progrès récents pour les rendre moins intrusives, les méthodes actives restent difficilement utilisables car éblouissant bien souvent l'utilisateur. Nous nous intéressons donc aux méthodes passives.

Une catégorie de modélisations des formes très populaires consiste au calcul de l'enveloppe visuelle à partir des silhouettes de l'objet. Cette enveloppe visuelle donne une relativement bonne approximation de la géométrie de la scène. Son calcul est robuste

aux bruits d'entrée et il est assez rapide (moins d'une seconde pour une scène filmée depuis une dizaine de caméras). L'amélioration de la précision de la géomètrie extraite nécessite l'utilisation d'informations photométriques. Malgré des améliorations récentes grâce à l'utilisation du GPU, ces approches de stéréo multi-vue sont très coûteuses et nécessitent plusieurs dizaines de seconde par trame. Ces approches, même parallélisées, sont donc difficilement utilisables aujourd'hui dans des systèmes d'interaction.

Le modèle d'enveloppe visuelle semble donc être bien adapté aux systèmes d'interaction. Trois grandes familles d'approches ont été proposées pour calculer ce modèle.

Les approches volumiques se basent sur la recherche de l'occupation de l'espace selon un découpage régulier. Ces méthodes présentent l'avantage d'avoir une complexité linéaire en le nombre de caméras, mais ne sont pas avantageuses en terme de coût pour obtenir des modèles de grande précision. Les représentations volumiques sont en outre très utiles dans certains cas, comme les interactions avec des simulations eulériennes comme les fluides.

Les approches surfaciques quant à elles cherchent à modéliser directement la surface de la scène. Grâce à de récents travaux, l'enveloppe visuelle polyédrique exacte peut être calculée à l'aide de calculs géométriques relativement simples. Ces approches permettent ainsi d'obtenir un maillage représentant la scène : ce type de données est très utile pour l'affichage du modèle ou son utilisation dans des simulations basées sur les maillages. En revanche il est assez difficile à partir de ces modèles de déterminer l'occupation de l'espace en un point.

Enfin une approche hybride a été présentée et permet d'obtenir simultanément une information sur l'occupation de l'espace et une information sur la surface de la scène sous la forme d'un maillage.

Modélisations surfaciques distribuées 4

4.1 Introduction

Nous présentons dans ce chapitre deux algorithmes distribués de calcul d'enveloppes visuelles. Ceux-ci sont basés sur les méthodes séquentielles de Franco et Boyer pour la modélisation surfacique [Franco 03] et hybride [Boyer 03]. Bien que déjà relativement rapides, ces algorithmes ne permettent pas une exécution temps réel. Pour cela, nous avons étudié, dans le cadre de cette thèse, la parallélisation des calculs mis en jeu. Comme nous l'avons expliqué à la section 2.2.3, cette distribution des calculs internes – la parallélisation par trame – permet d'améliorer les performances de la modélisation en latence comme en débit.

Avant de décrire ces algorithmes, nous présentons quelques définitions, sur la structure de l'enveloppe visuelle, utiles pour leur description. Ces définitions sont représentées sur la figure 4.1. Rappelons que la scène est filmée par N caméras *a sténopé* dont la calibration est connue. Nous disposons d'autre part pour chaque image de sa silhouette sous forme binaire et vectorisée. Nous appellerons *vecteurs de contour* les arêtes des contours polygonalisés, renforçant ainsi la notion d'orientation qui leur est liée permettant d'identifier l'intérieur de la silhouette. La *ligne de vue* associée à un point P d'une image correspond à l'ensemble des points de l'espace se projetant en ce point. Avec le modèle de caméra *a sténopé*, cela équivaut à la ligne passant par le point P et le centre de la caméra associée à cette image. Le *cône de vue* lié à la caméra i, correspond à l'ensemble des

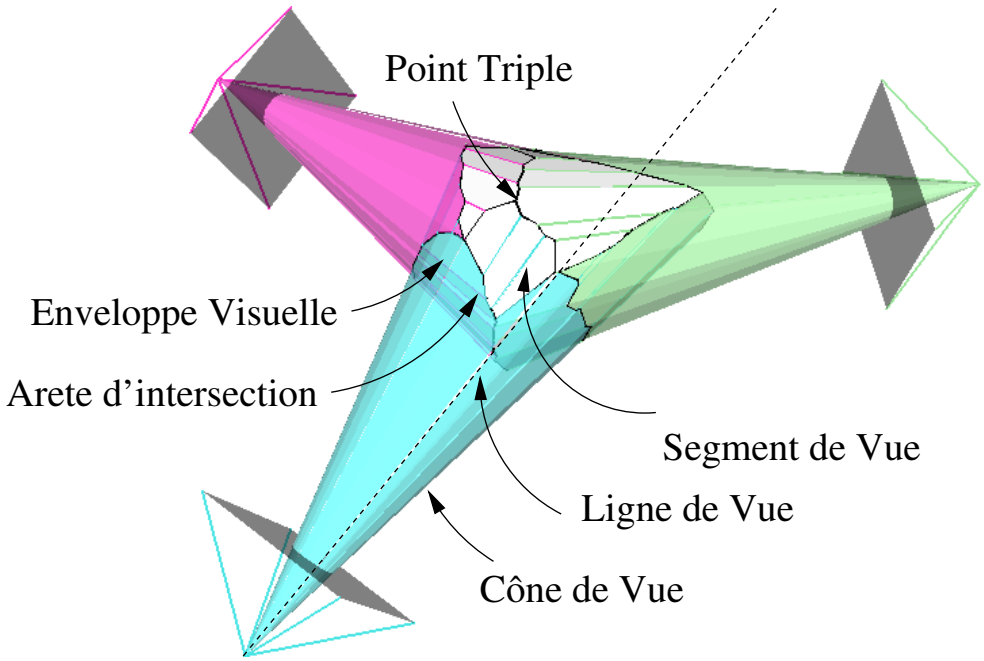


Figure 4.1 *Enveloppe visuelle polyèdrique obtenue à partir de trois silhouettes vectorisées et les primitives associées.*

lignes de vue associées aux points à l'intérieur de la silhouette de la caméra i. Autrement dit, le cône de vue est l'ensemble des points de l'espace dont la projection est à l'intérieur de la silhouette. Remarquons que les contours étant polygonalisés, le cône de vue est un volume délimité par des *facettes triangulaires infinies* associées aux vecteurs des contours. L'enveloppe visuelle correspond alors à l'intersection des cônes de vue associés aux caméras. L'enveloppe visuelle est ainsi un polyèdre. Les arêtes correspondent soit à des segments de ligne de vue (*segments de vue*) soit à des intersections entre deux facettes de cône de vue provenant de caméras différentes (*arêtes d'intersection*). Les sommets de l'enveloppe visuelle correspondent à des extrémités de segments de vue ou à des *points triples*, intersections de facettes issues de trois caméras différentes.

Nous présentons d'abord dans la section 4.2 la méthode de calcul des segments de vue, étape commune aux deux méthodes. Nous détaillons ensuite ces deux algorithmes permettant de retrouver l'enveloppe visuelle à partir de ces segments de vue. La section 4.3 présente la modélisation hybride qui cherche à approximer l'enveloppe visuelle à l'aide d'une tétraédrisation de l'espace. La section 4.4 présente la modélisation surfacique qui calcule l'enveloppe visuelle polyédrique exacte en retrouvant les arêtes d'intersection et les points triples. La figure 4.2 présente un aperçu des différentes étapes mises en jeu pour ces algorithmes. Notre contribution dans ce chapitre concerne la parallélisation de chacune de ces étapes, les versions séquentielles étant le fruit de travaux précé-

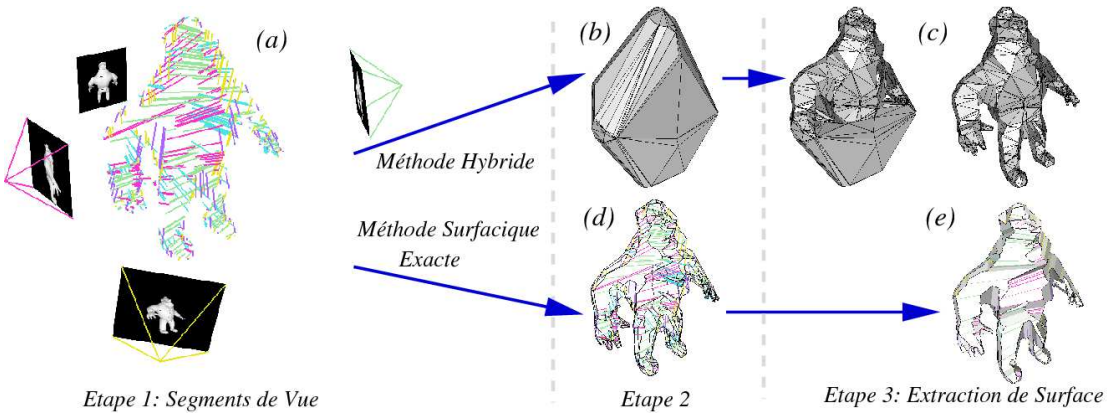


Figure 4.2 *Schéma illustrant le déroulement des différentes techniques de modélisation de l'enveloppe visuelle que nous parallélisons. (a) Les segments de vue sont calculés à partir des silhouettes vectorisées. (b) La méthode hybride calcule une triangulation de Delaunay des points extrêmes des segments de vue. (c) Chaque tétraèdre est éliminé s'il se projette en dehors d'une silhouette. (d) La méthode surfacique exacte calcule la géométrie des intersections des cônes de vue, afin de finir le calcul du maillage de l'enveloppe visuelle. (e) Les facettes sont extraites de cette dernière représentation finalisant le modèle de l'enveloppe visuelle.*

dents [Boyer 03, Franco 03]. Ce travail a été réalisé en collaboration avec Jean-Sébastien Franco, auteur des versions séquentielles de ces algorithmes.

4.2 Calcul des segments de vue

Le calcul des segments de vue est une étape initiale commune pour le calcul de l'enveloppe visuelle pour les deux méthodes. Nous présentons dans un premier temps l'algorithme séquentiel développé par Franco *et al.* [Franco 05] puis le travail de parallélisation que nous avons effectué.

L'algorithme consiste à calculer pour chaque ligne de vue issue d'un sommet d'un contour dans les images, sa contribution à la surface de l'enveloppe visuelle. On recherche donc l'intersection de cette ligne de vue avec les cônes de vue issus des autres silhouettes. Chaque intersection avec un de ces cônes de vue prend la forme d'une liste d'intervalles le long de cette ligne de vue (voir figure 4.3). Ces intervalles issus des différents points de vue sont ensuite fusionnés afin d'obtenir au final une primitive – des segments – appartenant à l'intersection de tous les autres cônes et donc à la surface de l'enveloppe visuelle polyédrique. Franco *et al.* [Franco 05] proposent différentes formules de mise à jour des intervalles ; dans notre cas nous avons choisi d'intersecter ces intervalles. L'objet doit donc être intégralement visible dans chaque image. Voici l'algorithme détaillé :

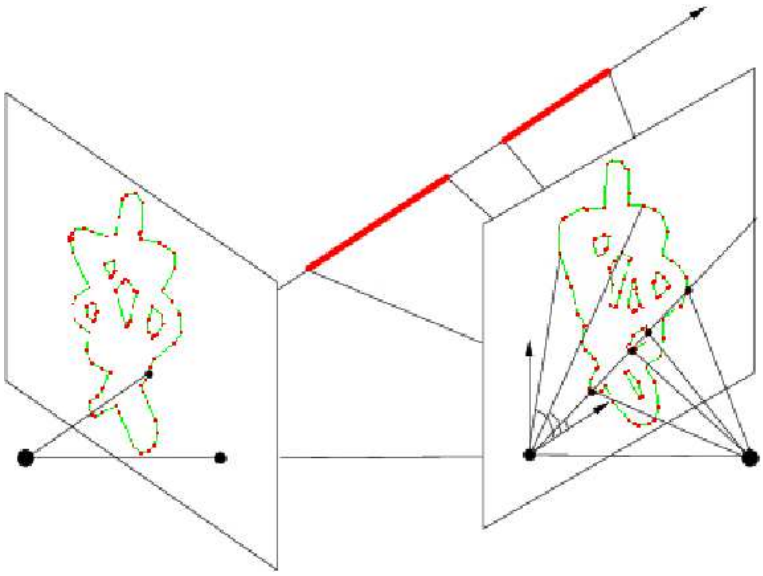


Figure 4.3 *Intervalles de contribution d'une caméra à l'enveloppe visuelle (en rouge) le long de la ligne de vue.*

Algorithme 1 Calcul des segments de vue

1: **pour chaque** sommet S d'un contour d'une image i : **faire**
2: **pour chaque** image k telle que $k \neq i$: **faire**
3: **calculer** la ligne épipolaire l de S dans l'image k,
4: **calculer** les intersections de l avec tous les contours de l'image k,
5: **mettre à jour** les intervalles le long de la ligne de vue associée à S.
6: **calculer** les points $3D$ délimitant les intervalles le long de la ligne de vue associée à S.
7: **fin pour**
8: **fin pour**

Remarquons que les calculs d'intersections ne se font pas directement en $3D$ mais plus judicieusement dans les images grâce à la géométrie épipolaire (voir figure 4.3). La complexité de cet algorithme dépend du carré du nombre de caméras, du nombre de contours par caméras et du nombre de sommets.

Pour paralléliser cet algorithme sur p processeurs, deux solutions se présentent selon la boucle sur laquelle porte la distribution.

Distribution des lignes de vue

La première solution consiste à distribuer la première boucle, c'est à dire à répartir

l'ensemble des sommets en p groupes et en associant un groupe à chaque processeur. Cela est possible du fait que chaque ligne de vue peut être traitée indépendamment des autres. Cela implique cependant que chaque processeur doit connaître les contours des silhouettes de chaque caméra. Dans ce schéma de distribution, il est important d'équilibrer au mieux les calculs entre les différentes ressources en choisissant de manière adéquate la répartition des sommets des contours. En supposant les processeurs identiques et sans *a priori* sur les calculs liés à chaque ligne de vue, nous devons distribuer le même nombre de sommets de contours à chaque processeur. Chaque ligne de vue ne nécessite cependant pas exactement la même quantité de calculs, engendrant potentiellement un déséquilibre. Celui-ci reste cependant relativement léger : ceci est du au fait que chaque processeur traite plusieurs dizaines de lignes de vue, moyennant ainsi les variations.

Parallélisation par anneau

La seconde solution consiste à paralléliser la seconde boucle. Pour simplifier la description nous considérerons que nous avons un processeur par caméra et que celui-ci est dédié aux intersections des segments de vue avec les contours de cette caméra. Chaque ligne de vue doit donc passer de processeurs en processeurs pour s'intersecter avec les différents points de vue. Cette solution consiste en une stratégie en anneau. Chaque processeur i reçoit des segments de vue du processeur précédent $i-1$, il les intersecte avec son cône de vue et réémet les intervalles mis à jour au processeur suivant $i+1$. Cet algorithme nécessite donc p étapes communications/calculs dans le modèle BSP [Valiant 90]. Ceci est particulièrement désavantageux si on considére la latence des communications. De plus la quantité d'informations véhiculées sur le réseau s'avère supérieure à celle de la solution précédente.

Cette parallélisation par anneau est ainsi moins efficace que la distribution des lignes de vue. Nous choisissons donc d'utiliser la première stratégie. Notons qu'à la fin de cette étape, l'ensemble des segments de vue est partitionné sur les différents processeurs et qu'il est nécessaire de les regrouper par des communications réseau.

4.3 Modélisation par tétraédrisation

Les segments de vue fournissent un premier échantillonage de la surface de l'enveloppe visuelle. En particulier, en considérant les extrémités de ces segments nous obtenons un nuage de points de cette enveloppe visuelle. Cet ensemble de points est utilisé par Boyer *et al.* [Boyer 03] pour obtenir une approximation de l'enveloppe visuelle par une méthode hybride. L'espace est découpé en cellules grâce à une tétraédrisation de Delaunay. Le découpage obtenu présente l'avantage d'être adapté à la forme de l'enveloppe visuelle en épousant sa surface. L'algorithme détermine ensuite pour chaque cellule son

appartenance ou non à l'enveloppe visuelle : la cellule est considérée comme faisant partie de l'objet si sa projection dans chaque image est incluse dans la silhouette correspondante. L'approximation de cette modélisation est d'autant plus faible que le nuage de points est dense. La surface est très facilement extraite en identifiant les facettes frontières qui délimitent deux cellules, l'une à l'intérieur et l'autre à l'extérieur de l'enveloppe visuelle.

Les performances de cet algorithme sont principalement limitées par la tétraédrisation de Delaunay. Cette dernière nécessite environ 80 ms pour traiter 2000 points sur un processeur Opteron 2 GHz. L'étape d'identification des cellules est très rapide car le nombre de cellules à tester est relativement faible, entre 6000 et 10000. Ce nombre de cellules est en particulier beaucoup plus faible que dans le cas d'une modélisation volumique pour une précision équivalente (pour une précision de $2cm$ ces méthodes nécessitent environ un million de cellules).

Pour améliorer les performances de cette méthode nous pouvons chercher à utiliser plusieurs processeurs pour effectuer plus rapidement les calculs. Cependant la parallélisation de la tétraédrisation de Delaunay est un problème difficile. En effet, même si le problème est généralement assez localisé, il existe des cas où le traitement d'un point entraîne la modification complète de la structure de données. Ceci est particulièrement vrai en trois dimensions comme le montre les récents travaux de thèse de Joseph Kohout [Kohout 05]. Plusieurs équipes proposent des algorithmes distribués de tétraédrisation de Delaunay [Cignoni 95, Kohout 03]. Ces travaux montrent tous que lorsque le nombre de points est inférieur à 10000 les performances ne sont qu'au mieux doublées malgré l'utilisation de plusieurs processeurs. La parallélisation de la tétraédrisation de Delaunay dans nos conditions n'est donc pas efficace. De plus une grande partie des performances de ces algorithmes repose sur la qualité des structures de données et sur leur implantation. Ainsi les implantations séquentielles optimisées comme Qhull [Qhull] sont généralement beaucoup plus rapides que les implantations parallèles car ces dernières reposent sur des structures de données moins optimisées. La parallélisation par trame de la tétraédrisation de Delaunay n'apporte donc que très peu de bénéfice. En revanche notons que des traitements à $30Hz$ sont envisageables en utilisant le parallélisme de flux avec plusieurs unités de calcul, la tétraédrisation de Delaunay étant statiquement calculable. Dans ce cas la latence de l'application restera inchangée et supérieure à $80ms$.

L'identification des cellules internes peut quant à elle grandement bénéficier du parallélisme. En effet le traitement de chaque cellule est indépendant. Les cellules peuvent donc être réparties sur les différents processeurs pour être traitées. Remarquons alors que chaque processeur doit connaître l'ensemble des silhouettes sous la forme binaire (beaucoup plus efficace pour cette tâche que la forme vectorisée). Comme dans le cas des segments de vue nous pouvons imaginer une parallélisation avec une stratégie par anneau évitant ainsi une diffusion des silhouettes binaires. Cependant cette stratégie nécessite un grand nombre de communications pour véhiculer les cellules entre les différents proces-

seurs. Ces données sont généralement beaucoup plus volumineuses que les silhouettes qui se compressent très bien comme nous l'avons dit. Ceux-ci rend cette stratégie de parallélisation par anneau moins efficace que la répartition des cellules.

4.4 Modélisation surfacique

La modélisation hybride ne fournit qu'une approximation de l'enveloppe visuelle. Pour obtenir l'enveloppe visuelle exacte, il nous faut utiliser une approche différente et retrouver les arêtes et sommets manquants, c'est-à-dire les arêtes d'intersection et les points triples.

Pour cela nous nous appuyons sur la méthode développée par Franco *et al.* [Franco 03]. Celle-ci consiste à rechercher les arêtes d'intersection à partir des informations préalablement calculées sur les segments de vue. Nous ne détaillons pas complètement dans cette thèse l'algorithme mais donnons les éléments essentiels pour sa compréhension. Pour plus de détails, nous renvoyons le lecteur intéressé à la thèse de Jean-Sébastien Franco [Franco 05]. L'algorithme repose principalement sur le fait que chaque sommet de l'enveloppe visuelle polyédrique est trivalent dans les situations réelles. En effet ces sommets correspondent à l'intersection de facettes triangulaires infinies. Or l'intersection de 4 plans dans l'espace est numériquement improbable (cas dégénéré qui disparait lors de l'ajout d'un bruit infime). Les sommets de l'enveloppe visuelle sont donc l'intersection de trois facettes triangulaires infinies et sont donc trivalents. Un point très important qui en découle est le fait que ces trois triangles – et donc les trois vecteurs de contours associées – identifient de façon unique un sommet. La connaissance de ces trois facettes permet aussi de déterminer les directions des trois arêtes du sommet associé (directions données par les intersections entre deux de ces facettes). Pour chaque sommet, nous cherchons les extrémités des arêtes non résolues partant de ce sommet. Pour cela nous calculons l'intersection de la demi-droite associée à cette arête – d'origine ce sommet et de direction celle de l'arête – avec la surface de l'enveloppe visuelle. Cette recherche se fait en pratique dans les images $2D$ par intersection avec les contours. L'extrémité peut soit être un point déjà connu, soit correspondre à un nouveau point triple. Connaissant les trois facettes générant cette extrémité (deux issues de l'arête et la troisième issue de l'intersection avec un vecteur de contour), il est très facile d'identifier ce point avec exactitude. En particulier cela permet de ne pas avoir recours à des heuristiques liées à la proximité entre l'extrémité trouvée et les points déjà visitées. Ce parcours du maillage de l'enveloppe visuelle polyèdrique est effectué jusqu'à ce qu'il n'y ait plus d'arêtes non résolues.

Cet algorithme présente de très nombreux avantages dont une grande robustesse aux cas dégénérés et une grande rapidité par rapport à d'autres méthodes surfaciques. De plus

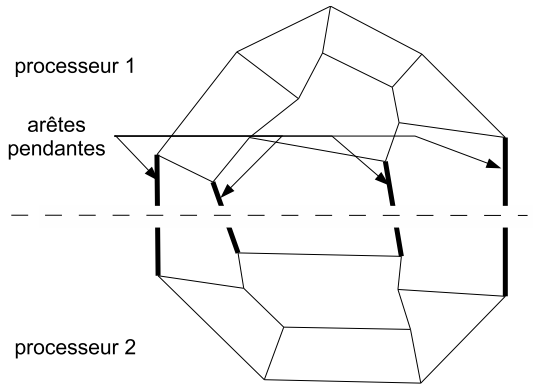


Figure 4.4 *Découpage de l'espace pour la parallélisation du calclul des arêtes d'intersection manquantes. Chaque processeur traite les arêtes de sa zone. Les arêtes franchissant une frontière entre deux zones, baptisées arêtes pendantes (en gras sur la figure), sont identifiées et fusionnées ultérieurement.*

la solution retrouvée est la solution exacte par rapport aux silhouettes vectorisées et non plus une approximation. En pratique cette étape nécessite environ $100ms$ sur un Xeon 3 GHz pour modéliser une personne filmée depuis 8 caméras avec environ 120 sommets de contours par silhouette. Pour améliorer ces performances nous allons chercher à distribuer ces calculs sur un ensemble de p processeurs. Remarquons au passage que cette étape est aussi statiquement calculable et qu'une parallélisation par flux est envisageable. Cependant il est toujours plus intéressant d'effectuer une parallélisation par trame, cette dernière améliorant simultanément le débit et la latence.

La parallélisation de cet algorithme n'est pas aussi simple que celle du calcul des segments de vue où chaque tâche était indépendante. La difficulté provient du fait que cet algorithme nécessite à chaque visite d'un sommet la mise à jour de la structure du maillage en cours créant ainsi des dépendances.

Une première solution pour effectuer cela est de supposer que nous disposons d'un ordinateur à mémoire partagée, c'est à dire que les processeurs partagent une zone mémoire dans laquelle peut être stockée la structure du maillage. Ainsi chaque processeur peut accéder à cette structure. Il est nécessaire alors de protéger cette dernière contre les accès concurrents : si deux processeurs veulent accéder au même sommet de l'enveloppe visuelle, il est nécessaire de donner la priorité à l'un avant d'autoriser le traitement du second. Sans ce mécanisme, l'exécution de l'algorithme risque d'être incorrect. Ces protections s'avèrent très coûteuses d'autant que la localité des calculs est difficilement garantie lors du parcours du maillage. Un autre inconvénient de cette stratégie est la nécessité d'obtenir une mémoire partagée. Ceci est relativement aisé

jusqu'à 4 processeurs mais devient assez coûteux au-delà voire très cher pour plus de 16 processeurs.

Pour s'affranchir de cette structure centralisée il est nécessaire d'imposer plus de localité au parcours du maillage. Ainsi nous proposons de partitionner l'espace en p zones, chaque processeur étant responsable d'une zone qui lui est attribuée. Chaque processeur effectue le parcours au sein de sa zone. Dès qu'il détecte une arête dont l'extrémité sort de cette zone, il l'enregistre en tant qu'*arête pendante* (voir figure 4.4). Une fois chaque zone parcourue, l'ensemble des points triples, arêtes d'intersection et arêtes pendantes sont regroupés sur un processeur central responsable de la fusion de ces données. Tout d'abord l'ensemble des points triples est ajouté comme nouveaux sommets dans une structure globale. Puis chaque arête d'intersection est ajoutée dans cette structure. Notons que ceci est possible grâce au fait que chaque sommet de l'enveloppe visuelle peut être identifié grâce aux trois facettes triangulaires infinies qui le définissent. Il est donc important lors du calcul des arêtes d'intersection de mémoriser pour chaque extrémité ces informations.

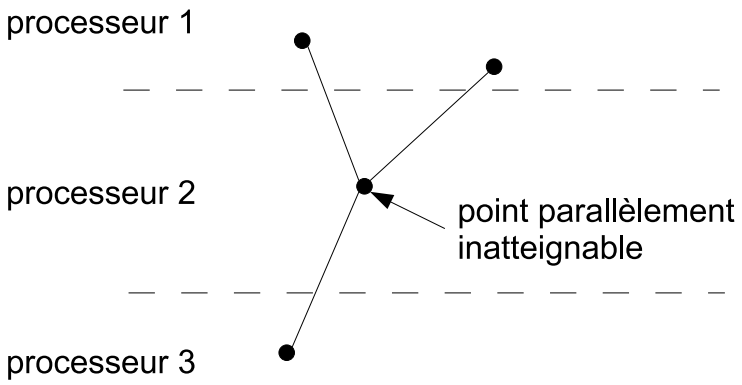


Figure 4.5 *Point triple parallèlement inatteignable. Le traitement de ces points doit être réalisé séquentiellement après fusion des données parallèles.*

Pour les arêtes pendantes la situation est un peu plus complexe. Si les deux extrémités sont présentes dans la structure globale, alors le traitement est identique à une arête d'intersection en prenant soin de ne pas l'ajouter deux fois. En revanche il existe des situations où une des extrémités n'est pas présente dans la structure globale. Ce cas correspond à un point triple appartenant à une zone j mais dont le parcours interne de cette zone n'a pas permis de l'identifier (voir figure 4.5). Nous appellerons un tel point un point parallèlement inatteignable. Dans ce cas, le point manquant est créé et l'arête peut être ajoutée comme précédemment. Il reste enfin à vérifier pour tous ces points triples parallélement inatteignables que leurs trois arêtes aient été correctement retrouvées. Dans le

cas contraire, un parcours séquentiel est effectué pour retrouver le maillage final complet. Pour équilibrer la charge de travail entre les processeurs, nous attribuons à chacun une zone comportant le même nombre d'extrémités de segments de vue.

Le maillage calculé, la dernière étape consiste à identifier chaque facette de l'enveloppe visuelle. Chaque facette est un sous-ensemble d'une facette triangulaire infinie associée à un vecteur de contours. Franco *et al.* proposent de rechercher ces sous-ensembles pour chaque facette triangulaire infinie en parcourant les sommets qui lui sont liés. En utilisant l'orientation préservée du maillage il est possible à partir d'un sommet de suivre les arêtes délimitant la facette. La parallélisation de cette étape s'effectue en partitionnant l'ensemble des vecteurs de contours, le traitement de chaque facette triangulaire infinie étant indépendante. Là encore ce découpage est réalisé de telle sorte que chaque processeur ait le même nombre de facettes à traiter pour équilibrer la charge entre eux.

4.5 Résultats

Pour mesurer l'influence de la parallélisation sur les performances de la modélisation nous avons implanté ces algorithmes distribués. Cette implantation est détaillée dans le chapitre 6. Etant donné que la tétraédrisation de Delaunay n'est pas parallélisée, nous nous concentrons sur la modélisation surfacique exacte. Les tests ont été effectués sur une grappe de PCs Xeon cadencés à 2.66 GHz. Nous avons surtout souhaité étudier la capacité de ces algorithmes à passer à l'échelle lorsque le nombre de points de vue augmente. Nous avons utiliser un modèle synthétique, baptisé Al Capone, pour générer un grand nombre de silhouettes selon différents points de vue. Ceci nous permet de tester la distribution des calculs avec un très grand nombre de caméras. Nous nous intéressons ici au problème de la latence. Le problème du débit peut être résolu en allouant suffisemment d'unités de traitement lors de la parallélisation par flux en utilisant le fait que chaque étape est statiquement calculable.

Le modèle synthétique est un personnage humanoïde, cela pour se rapprocher le plus possible de notre contexte de système d'interaction. Il présente une complexité similaire à un personnage réel dans notre contexte : sa projection dans les images engendre des silhouettes polygonales comportant environ 130 sommets. La figure 4.6 présente les temps de modélisation obtenus pour 16, 25 et 64 points de vue. Notons que ces temps ne prennent pas en compte les temps de traitement d'image. La parallélisation de l'algorithme permet de réduire ce temps, et donc la latence, de manière significative (presque un ordre de grandeur). Avec 12 points de vue et 8 processeurs, le temps de modélisation est inférieur à 100 ms, une latence acceptable pour l'interactivité du système.

La figure 4.7 présente les accélérations associées, c'est-à-dire le rapport entre le temps de la modélisation distribuée sur p processeurs comparé au temps de la version séquentielle. Le facteur d'accélération est supérieur à la moitié du nombre de processeurs utilisés juqu'à 9 processeurs pour 12 points de vue, 14 processeurs pour 25 points de vue et plus

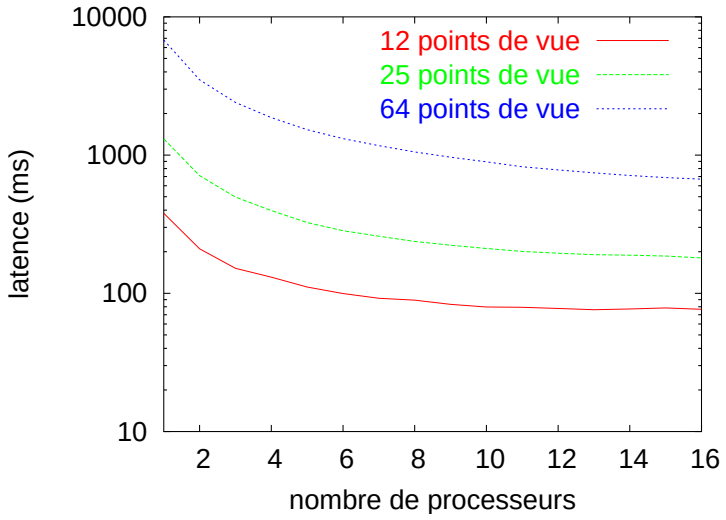


Figure 4.6 *Graphe logarithmique des latences pour le personnage de synthèse.*

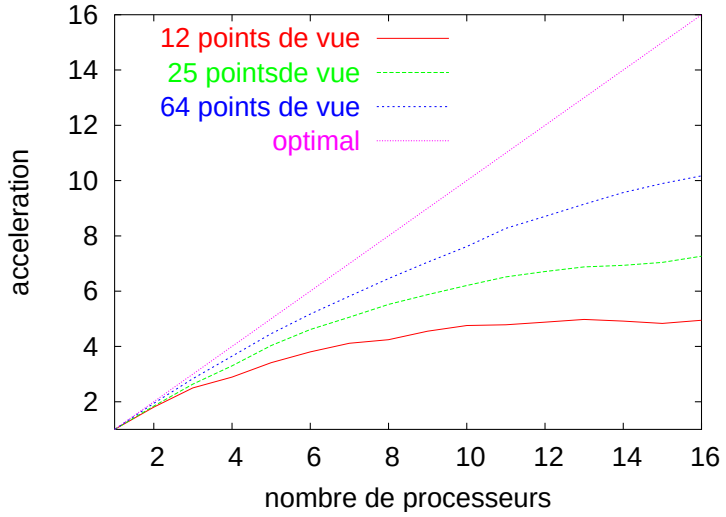


Figure 4.7 *Accélérations pour le personnage de synthèse.*

de 16 processeurs pour 64 points de vue. Idéalement l'accélération devrait correspondre à p, le nombre de processeurs utilisés. Cette différence s'explique par le temps requis par les communications des données entre processeurs et les attentes liées au fait que tous les processeurs ne terminent pas leurs calculs ensemble de manière parfaitement synchrone. De plus la seconde étape de l'algorithme – le calcul des arêtes d'intersection et des points triple – comporte des calculs redondants (chaque arête pendante est calculée en double) et nécessite dans certains cas (points parallèlement inatteignables) des calculs séquen-

tiels pour compléter le maillage. Ces deux phénomènes rendent la parallélisation moins efficace. Plus le nombre de processeurs est grand, plus le nombre d'arêtes pendantes augmente et plus les cas de points parallèlement inatteignables deviennent fréquents. Ainsi lorsque le nombre de processeurs augmente beaucoup, les performances ne s'améliorent que très lentement (accélération quasi constante) et peuvent même diminuer lorsque le nombre de processeurs devient trop grand.

4.6 Bilan

Nous avons présenté dans ce chapitre, la parallélisation interne de chacune des étapes impliquées dans la modélisation surfacique exacte. Une version hybride a aussi été considérée mais avec moins de succès en raison des difficultés de parallélisation de la tétraédrisation de Delaunay. Cette distribution des calculs internes sur plusieurs processeurs permet de réduire la latence et par là même d'améliorer le débit. Elle permet aussi d'envisager une augmentation du nombre de caméras dans l'optique d'améliorer la précision des modèles reconstruits tout en maintenant des performances temps réel. Les performances de cette modélisation distribuée dans le contexte réel sont données dans le chapitre 7 page 77.

Ces travaux montrent que l'utilisation de techniques de parallélisation relativement simples permet cependant d'obtenir de bons résultats. Une discussion sur l'efficacité de la parallélisation des algorithmes de vision par ordinateur est donnée dans la conclusion générale à ces travaux de thèse au chapitre 11 page 115.

Certaines améliorations sont envisageables. Tout d'abord nous n'avons considéré ici que des répartitions statiques des travaux sur les processeurs, pouvant provoquer des déséquilibres de charge entre processeurs. L'utilisation de mécanisme adaptatif comme celui qui sera présenté au prochain chapitre permettrait sans doute de résoudre ce problème.

En ce qui concerne la seconde étape, c'est-à-dire le parcours du maillage de la surface, plusieurs modifications pourraient grandement améliorer l'efficacité de la parallélisation. Le découpage en tranches pourrait par exemple être avantageusement remplacé par un découpage plus adéquat cherchant à minimiser les frontières entre les zones des différents processeurs : ceci permet alors de minimiser le nombre d'arêtes traversant les frontières et devant être traitées séquentiellement. Les coûts de communication de ces arêtes peuvent aussi être divisés par deux : en effet il suffit de ne transmettre que les arêtes pendantes allant d'une zone i à une zone j où $i < j$.

Octree parallélisé 5

Comme nous l'avons vu dans l'état de l'art au chapitre 3 page 29, la modélisation $3D$ peut se faire par une approce volumique ou surfacique. Le modèle surfacique, présenté au chapitre précédent, est bien adapté pour le rendu et peut garantir aucune perte d'information par rapport aux silhouettes d'entrée. Les approches volumiques donnent accès à une information d'occupation de l'espace utile pour certaines applications utilisant elles aussi un modèle volumique comme l'écoulement de fluide (voir le chapitre 7 page 77).

Bien que fournissant une telle information volumique, la modélisation hybride ne s'avère pas bien adaptée pour ce type d'applications : elle ne permet pas d'obtenir en temps constant l'occupation d'un point de l'espace. Pour cela, il est nécessaire d'utiliser une modélisation volumique basée sur un découpage régulier de l'espace. Une manière efficace d'effectuer une telle modélisation est d'utiliser une structure d'octree pour concentrer les calculs sur les zones importantes.

La parallélisation de ce type d'approches permet d'obtenir des modèles plus précis en accédant, pour un temps de calcul équivalent, à un niveau de détail plus fin. Ces algorithmes présentent des enjeux spécifiques et nécessitent l'utilisation de stratégies de parallélisation plus fines que celles utilisées dans le chapitre précédent pour la modélisation surfacique.

Nous décrivons tout d'abord l'algorithme séquentiel de modélisation par octree dans la section 5.1. Les enjeux de sa distribution ainsi que les principes de la parallélisation adaptative par vol de tâches sont détaillées dans la section 5.2. La section 5.3 détaille l'implantation de la parallélisation de cette modélisation distribuée à base d'octree. Enfin les résultats, principalement sur le plan des performances, sont présentés dans la section 5.4. Ces travaux ont été réalisés en collaboration avec Luciano Soares et Jean-Louis Roch.

5.1 Modélisation par octree

Cette modélisation, proposée par Szeliski [Szeliski 93], consiste en la recherche de l'occupation de l'espace grâce à un découpage selon une structure hiérarchique, un *octree*. La scène est représentée sous la forme d'un arbre dont chaque noeud représente un parallélépipède, appelé cube par abus de language. La racine de l'arbre correspond à la boîte englobante de la scène. Les fils d'un noeud correspondent aux 8 cubes de taille moitié obtenues par division du cube parent. La modélisation détermine pour chaque cube son type :

- *plein* si il est intégralement contenu dans l'enveloppe visuelle ;
- *vide* si il est intégralement à l'extérieur de l'enveloppe visuelle ;
- *indéterminé* si il contient une frontière de l'enveloppe visuelle.

Les cubes pleins ou vides correspondent aux feuilles de l'arbre.

La modélisation part du noeud racine et détermine récursivement le type de chaque noeud : lorsque le cube associé est indeterminé, celui-ci est divisé en ses 8 fils et l'occupation de chacun est récursivement calculée.

Le calcul du type d'un cube est réalisé en discrétisant celui-ci à l'aide d'un ensemble de points. L'appartenance de chaque point à l'enveloppe visuelle est déterminée en projetant ceux-ci dans les images et en examinant leur appartenance aux différentes silhouettes. Le nombre de points testés est proportionnel à la surface projetée du cube dans les images, c'est-à-dire au carré de la taille d'un côté du cube. Le coût du calcul pour un noeud parent est ainsi moitié moindre que la somme des coûts pour ses fils. Remarquons de plus que le test d'un cube indéterminé est généralement beaucoup plus rapide, étant donné qu'il s'interrompt dès que deux points, l'un intérieur et l'autre extérieur à l'enveloppe visuelle, ont été trouvés.

Cette structure hiérarchique permet de concentrer les calculs sur les zones le nécessitant, c'est-à-dire à la surface de l'enveloppe visuelle. Le surcoût lié aux calculs des noeuds indéterminés est en général très inférieur aux gains apportés par cette hiérarchisation.

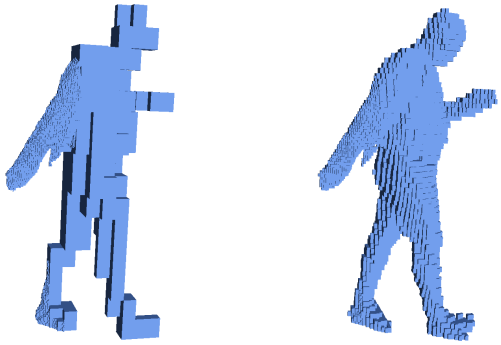
5.2 Parallélisation

5.2.1 Enjeux

La parallélisation de cet algorithme présente certains enjeux. Le premier est lié à la nécessité d'équilibrer la charge de chaque processeur. Le second est lié à la nécessité d'effectuer un parcours en largeur pour obtenir un niveau de détail homogène sur le modèle.

Equilibrage de charge

Lors de la parallélisation de la méthode surfacique nous avons partitionné l'ensemble des tâches et attribué une partie à chaque processeur. Si cette répartition statique s'avère efficace pour ces algorithmes, il est en tout autrement pour l'algorithme de modélisation par octree. En effet le coût de traitement d'un noeud peut varier énormément : il peut être très rapide dans le cas où il est plein ou très coûteux si il correspond à une zone frontière complexe (il comporte alors beaucoup de descendants à analyser). De plus il est très difficile de prédire à l'avance la complexité du traitement d'un noeud. Ainsi, avec un équilibrage statique, certains processeurs risquent de devoir attendre inutilement d'autres processeurs ayant reçu des tâches plus complexes. Ceci impose donc d'utiliser un mécanisme permettant de rééquilibrer la charge de chaque processeur en cours d'exécution.



(a) Parcours en profondeur (b) Parcours en largeur

Figure 5.1 *Exemple de modélisation avec un délai maximal lors d'un parcours en profondeur (à gauche) et un parcours en largeur (à droite). Lors d'un parcours en profondeur, la plupart des calculs se concentrent sur une zone (le bras droit ici). Un parcours en largeur permet d'obtenir des modèles de précision homogène.*

Parcours en largeur

Un intérêt de la modélisation par octree pour les systèmes interactifs est la possibilité d'imposer un contrôle du temps. L'algorithme explore l'arbre dans le temps imparti et calcule à la fin une approximation pour l'occupation des noeuds indéterminés ou non explorés (en testant un seul point central par exemple). Lorsque l'algorithme effectue le parcours en profondeur, la plupart du temps est passé dans l'analyse précise d'une zone bien particulière alors que le reste du modèle reste inexploré (voir figure 5.1). L'arbre doit donc être parcouru en largeur pour garantir un niveau de détail homogène. Ceci est particulièrement difficile à paralléliser. En effet un simple parcours en largeur localement pour chaque processeur ne permet pas de garantir que le modèle dans sa globalité comporte un niveau de détail homogène (un processeur pouvant aller plus vite que les autres par

exemple). Ce déséquilibre à l'instant t est caractérisé par $\sigma(t) = \max_{i \neq j} |L_i(t) - L_j(t)|$ où $L_i(t)$ représente la profondeur du noeud traité par le processeur i à l'instant t. Plus $\sigma(t)$ est petit, meilleure sera l'homogénéité du résultat avec idéalement $\sigma(t) = 0$.

5.2.2 Principe du parallélisme adaptatif par vol de tâches

Pour répondre à ces enjeux, une stratégie développée par la communauté du parallélisme consiste à effectuer une parallélisation adaptative par vol de tâches [Graham 66]. Le principe est relativement simple. L'ensemble des tâches à effectuer est attribué au démarrage de l'application à un processeur unique de façon équivalente à la version séquentielle. Lors de l'exécution, chaque processeur maintient une liste locale des tâches à traiter. Chaque processeur effectue tant qu'il peut les tâches présentes dans cette liste. Lorsque cette dernière devient vide – plus de tâches locales à effectuer – le processeur "vole" du travail à un autre processeur choisi au hasard. Remarquons que de nouvelles tâches peuvent être crées lors de l'exécution du programme : celles-ci sont ajoutées à la liste locale des tâches à effectuer du processeur effectuant ces créations.

Des travaux théoriques ont permis de démontrer les performances d'une telle stratégie en fonction du *travail* et de la *profondeur* de l'algorithme. Le travail correspond au nombre total d'opérations effectuées par l'algorithme : il équivaut donc au temps d'exécution séquentiel, noté T_1. La profondeur, notée T_∞, correspond au chemin critique, c'est-à-dire à la suite la plus longue de tâches dépendantes. Le temps d'exécution sur p processeurs en utilisant une stratégie par vol de tâches est alors borné par :

$$T_p \leq \frac{T_1}{p} + \mathcal{O}\left(T_\infty\right)$$

Ainsi, si la profondeur T_∞ est très petite comparée au travail T_1, le temps d'exécution parallèle est proche de l'optimal $\frac{T_1}{p}$. Dans cette situation le nombre de vols est en effet très faible limitant les surcoûts engendrés par les vols effectués et les transferts des données liées aux tâches.

5.2.3 Adéquation avec l'octree

L'utilisation d'une parallélisation adaptative par vol de tâches est particulièrement bien adaptée au problème de la modélisation par octree pour plusieurs raisons.

Tout d'abord, le nombre de tâches est nettement supérieur au chemin critique. Ce dernier correspond à la profondeur maximale de l'arbre (maximum 10 pour une modélisation de précision de l'ordre du pixel avec des images d'un million de pixels). Le nombre de noeuds est quant à lui exponentiel en la profondeur. Ainsi ce problème répond parfaitement aux critères théoriques énoncés précédemment permettant d'obtenir

une parallélisation efficace : $T_\infty \ll T_1$.

Deuxièmement la fonction de calcul du type d'un cube ne dépend que des coordonnées et de la taille de ce cube. En supposant que les silhouettes soient disponibles pour tous les processeurs, déplacer une tâche d'un processeur à un autre consiste donc simplement à transférer ces données du cube. Le surcoût lié au vol de tâches est donc très minime.

Enfin aucune dépendance entre tâches ne doit être prise en compte, celles-ci étant implicites dans le modèle. En effet une tâche n'est créée qu'au moment où celle-ci peut s'exécuter, c'est-à-dire lorsque le type du noeud parent a été calculé comme indéterminé. De plus une tâche ne dépend d'aucun résultat précédent. En particulier les résultats – sous la forme de la liste des cubes pleins – peuvent être stockés indépendamment et localement sur chaque processeur.

5.2.4 Contrainte relâchée de parcours en largeur

Une difficulté subsiste toutefois quant au parcours en largeur. En effet l'ordre d'exécution des tâches dans les systèmes par vol de tâches correspond habituellement à un parcours en profondeur. Modifier cette politique par un simple parcours en largeur sur chaque processeur ne fournit pas le résultat escompté comme nous l'avons précédemment expliqué : aucune borne sur $\sigma(t)$ n'est alors garantie. Fournir une telle garantie nécessite l'ajout de synchronisations entre les processeurs pour éviter ces déséquilibres. Placer des barrières de synchronisation après chaque profondeur de l'arbre permet de garantir $\sigma(t) = 0$ mais au détriment de l'efficacité de la parallélisation. Nous proposons d'introduire une contrainte relâchée, $\sigma(t) \leq 1$, permettant un recouvrement entre deux profondeurs successives. Ceci est très utile pour le parallélisme tout en maintenant un niveau de détail assez homogène. Pour cela les tâches sont organisées dans des listes différentes pour chaque profondeur. Ainsi un processeur finissant une liste de profondeur d ne pourra directement passer à la liste de profondeur $d + 1$ que si tous les processeurs ont terminé leurs tâches de niveau $d - 1$. Dans le cas contraire, il volera des tâches au processeur en retard.

5.3 Modélisation volumique distribuée : détails de l'implantation

Une tâche est représentée par un quadruplet (i, j, k, d) représentant le cube associé de coordonnées (i, j, k) et de profondeur d. En remarquant que les noeuds de très faible profondeur sont souvent indéterminés et pour pouvoir fournir du travail à chaque processeur au démarrage, l'algorithme débute à une profondeur n – habituellement

3 – générant ainsi dès le départ 8^n tâches. Cet ensemble de tâches est alors réparti aléatoirement sur les p listes locales à chaque processeur évitant ainsi les premiers vols (ceci peut être réalisé car les tâches sont indépendantes).

Pour obtenir les meilleures performances possibles et simplifier l'implantation, nous avons supposé que la modélisation s'exécute sur une seule machine à plusieurs processeurs possédant une mémoire partagée. Un thread principal est dédié au contrôle du temps et au chargement pour chaque trame des silhouettes dans cette mémoire partagée. Lorsque le temps imparti est écoulé, un signal est envoyé à tous les processeurs pour leur demander de terminer leurs calculs et d'envoyer leurs listes de cubes pleins calculés. Ces listes sont ensuite fusionnées par concaténation pour obtenir le modèle final.

L'algorithme a été implanté en C++ en utilisant les threads POSIX. Pour éviter le surcoût lié aux verrous de type sémaphore, nous avons utilisé des opérations atomiques comme "Compare et Echange". Ces opérations sont désormais supportées par la plupart des processeurs standards et permettent un gain de performance d'environ 20%. La gestion de la terminaison d'une profondeur d par l'ensemble des processeurs est réalisée par exemple à l'aide d'un compteur $C[d]$ initialisé à p (le nombre de processeurs) et décrémenté de façon atomique par chaque processeur lorsqu'il termine sa liste de tâches de profondeur d : ce compteur est alors nul lorsque tous les processeurs ont terminé cette profondeur.

Un thread par processeur est lancé pour les calculs en précisant à l´ordonnanceur du système d'affecter chaque thread à un processeur précis et de ne pas effectuer de migration lors de l'exécution.

5.4 Résultats

Cet algorithme a été testé sur une machine multi processeurs comportant 8 dual Core AMD Opteron cadencés à 2.2 GHz, 32 Go de mémoire et fonctionnant sous Linux. L'exécution sur des traitements hors-ligne nous permet de mesurer l'efficacité de cette parallélisation. L'exécution temps réel valide le mécanisme de contrôle du temps et l'intérêt des stratégies adaptatives pour les applications interactives.

5.4.1 Exécution hors-ligne

Nous avons d'abord testé notre modélisation distribuée par octree sur des données enregistrées et dont les silhouettes ont été préalablement extraites. Le premier jeu d'images, baptisé Ben, consiste en une personne filmée depuis 8 caméras chacune ayant une résolution de 780×582 (voir figure 5.2). Le second consiste en le même modèle synthétique qu'au chapitre précédent, Al Capone, grâce auquel nous avons généré 64

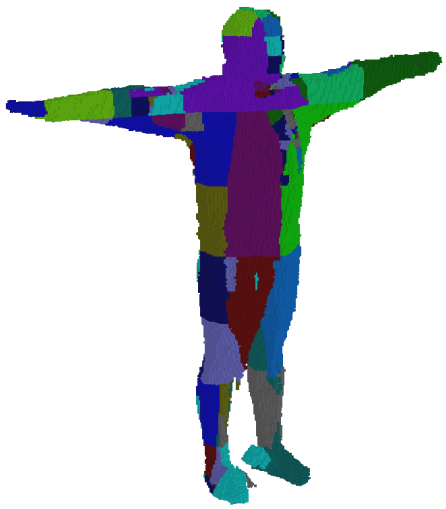


Figure 5.2 *Modélisation par octree de Ben avec une profondeur de* 8*. La couleur des cubes représente le processeur l'ayant calculé.*

images depuis différents points de vue avec une résolution de 300×300 (voir figure 5.3). Ce dernier jeu de données nous permet de valider notre approche pour un nombre très important de caméras. Remarquons que les temps mesurés dans cette partie comprennent le chargement des silhouettes et ont été obtenus en faisant une moyenne sur une centaine d'exécutions.

Le premier résultat concerne le surcoût introduit par la programmation parallèle. Pour cela nous avons comparé les temps d'exécution pour l'implantation séquentielle et celle parallèle sur un seul processeur. La différence de temps est relativement faible : elle est de l'ordre de 4% pour les données de Ben et de seulement 1.3% pour les données de synthèse.

Le second résultat concerne l'intérêt de la mise en place d'un système d'équilibrage de charge. Pour permettre de mesurer l'impact d'un tel mécanisme nous avons mesuré les temps d'exécution avec ou sans vol de tâches pour les données d'Al Capone et pour différents nombres de processeurs. Ces mesures sont présentées sur la figure 5.4. Nous pouvons aisément constater que l'utilisation d'un mécanisme adaptatif par vol de tâches est systématiquement plus efficace qu'une parallélisation par répartition statique des calculs.

Un autre résultat concerne le passage à l'échelle, c'est-à-dire l'efficacité du parallèlisme lorsque le nombre de processeurs augmente. La figure 5.5 (graphe du haut)

Figure 5.3 *Modélisation par octree de Al Capone avec une profondeur de* 5 *(à gauche) et* 7 *(à droite). La couleur des cubes représente le processeur l'ayant calculé.*

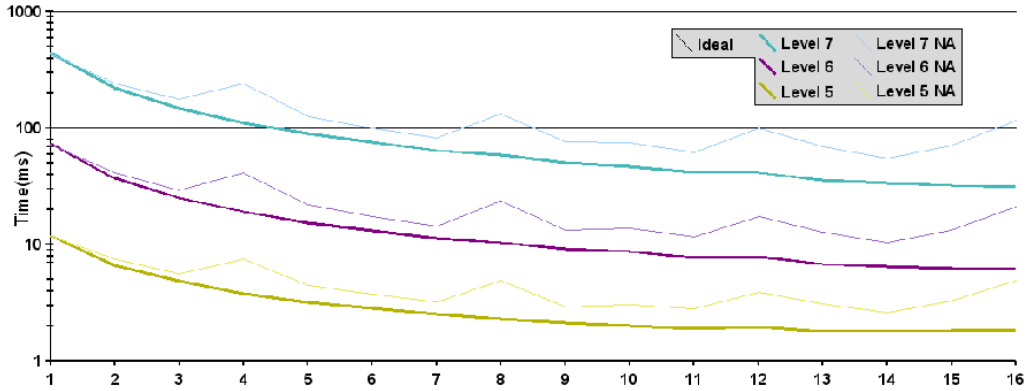


Figure 5.4 *Mesures du temps de modélisation pour les données de synthèse avec ou sans vol de tâches (NA = Non Adaptatif) et pour différents niveaux de détail. La parallélisation adaptative est systématiquement plus efficace.*

Jeu de données	Profondeur	Cubes explorés	Vols/processeurs
Ben	8	162799	42.59
Al Capone	7	44840	26.21

TAB. 5.1 *Nombre de cubes explorés et nombre de vols de tâches effectués par processeur pour les deux jeux de données et différentes profondeur.*

présente l'évolution du temps d'exécution en fonction du nombre de processeurs pour les données Ben avec différentes profondeurs. L'accélération, c'est-à-dire le rapport entre le temps séquentiel et le temps parallèle, est représentée pour les mêmes données sur le graphe du bas de la figure 5.5. Nous constatons que pour une profondeur de 8 l'utilisation de 16 processeurs permet d'aller 14 fois plus vite que la version mono-processeur. Le temps de modélisation passe donc de 234 ms en séquentiel à 17 ms. Le graphe des accélérations montre aussi que plus la profondeur maximale est grande, meilleure est la parallélisation. Ceci s'explique en partie par le fait que l'étape de chargement des images est séquentielle : plus le parcours de l'arbre est coûteux, moins ce temps de chargement intervient dans le coût total. Le tableau 5.1 présente le nombre moyen de vols de tâches effectués par processeur. Comme prévu ce nombre est très faible et explique l'efficacité de l'approche adaptative : le surcoût lié aux vols est relativement peu fréquent. De plus ce faible nombre de vols permet de conserver une grande localité des zones à traiter. Les figures 5.2 et 5.3 présentent les modèles reconstruits où la couleur d'un cube représente le processeur l'ayant traité : on remarque que de très grandes zones continues ont été traitées par un même processeur.

Enfin nous avons expérimenté le contrôle du temps sur les données Ben avec une profondeur maximale de 8 et un temps maximal de 30 ms. La figure 5.6 présente l'évolution du niveau de détail obtenu en fonction du nombre de processeurs utilisés. Avec un seul processeur, il n'est pas possible de terminer complètement la profondeur 5, rendant le modèle difficilement reconnaissable. Jusqu'à 8 processeurs, le contrôle du temps est utile : l'exécution se termine avant que l'ensemble des cubes de profondeur 8 n'aient été traités. Remarquons que le contrôle du temps est relativement efficace avec des temps légèrement supérieurs à 30 ms : ce dépassement est dû aux coûts de terminaison des cubes en cours de traitement (la valeur aberrante pour un seul processeur correspond au fait que l'on ne peut stopper les calculs des tâches crées au démarrage du programme). Avec 8 processeurs la profondeur 8 est alors totalement atteinte dans le temps imparti. Au-delà, les ressources supplémentaires de calcul servent à réduire le temps de calcul : le critère de terminaison n'est alors plus la contrainte du temps mais celle de la profondeur maximale.

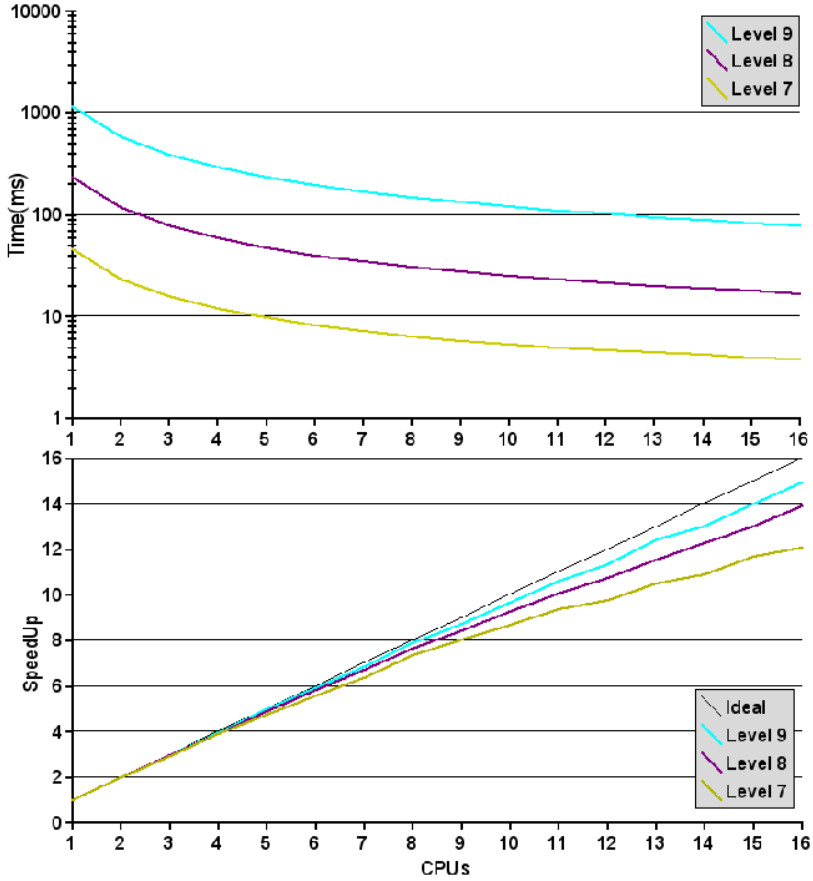


Figure 5.5 *Temps d'exécution et accélérations en fonction du nombre de processeurs pour différents niveaux de détail de la modélisation des données Ben.*

5.4.2 Exécution temps réel

Nous avons ensuite validé notre approche dans des conditions réelles de modélisation d'un acteur filmé par 5 caméras. La résolution de ces caméras est de 780×580. L'acquisition et l'extraction des silhouettes sont détaillées dans la partie III page 63. Les silhouettes une fois extraites sont transmises à la machine multi-processeur et sont chargées en mémoire par le thread principal de notre programme. Une fois la modélisation terminée, les résultats de chaque processeur, sous la forme de listes de cubes pleins, sont fusionnés et envoyés vers des machines permettant leur affichage. La plateforme expérimentale

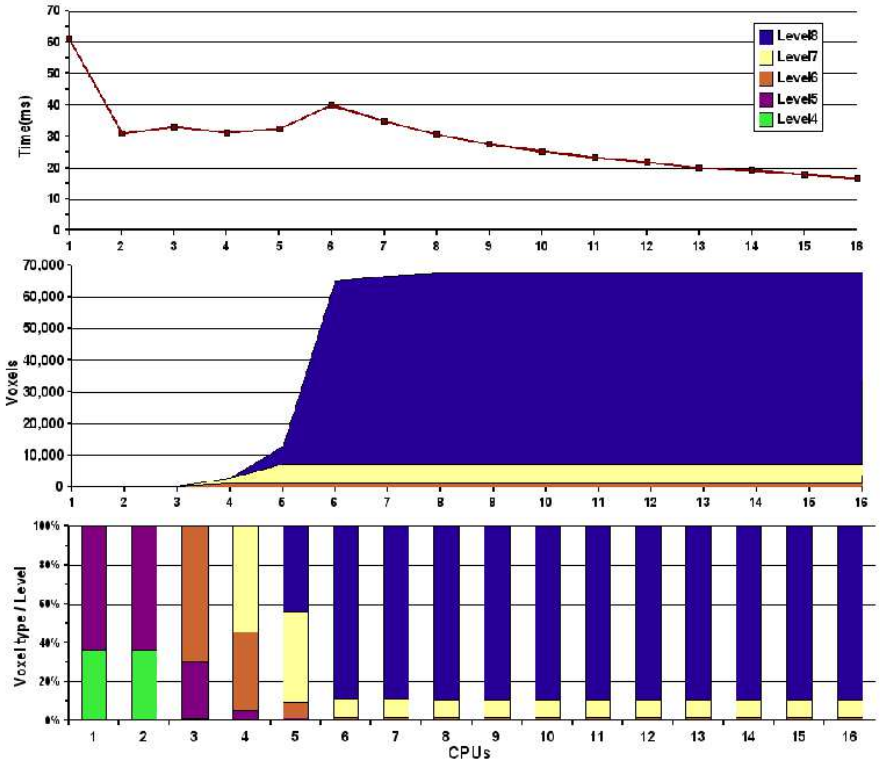
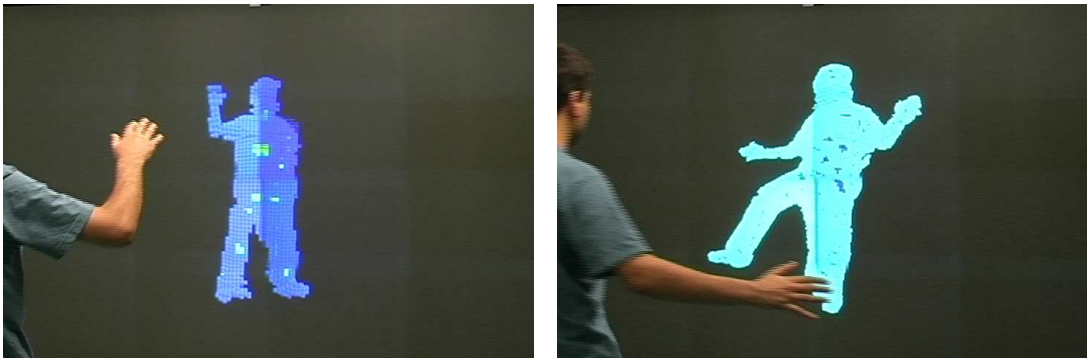


Figure 5.6 *Mesures sur la modélisation à partir des données de Ben avec une profondeur maximale de* 8 *et un temps maximal de* 30 *ms. Le graphe du haut représente le temps pris par cette modélisation. Le graphe du milieu correspond aux nombres de cubes pleins calculés. Le graphe du bas représente la proportion de cubes pleins pour chaque profondeur. La couleur bleu correpond à la profondeur* 8*, jaune pour* 7*, orange pour* 6*, violet pour* 5 *et vert pour* 4*).*

GrImage sur laquelle ont été réalisés ces tests est détaillée dans le chapitre 7 page 77.

La figure 5.7 présente le résultat de cette modélisation temps réel. Nous constatons aisément que la précision du modèle augmente avec la profondeur maximale atteinte. Alors que la modélisation est limitée à une profondeur de 6 avec un seul CPU, la parallélisation sur 16 processeurs permet d'atteindre une profondeur voisine de 8 équivalent à la précision d'une grille régulière de 256^3. Ceci permet en particulier d'obtenir une précision d'environ un centimètre et ainsi de détecter des détails tels que les doigts d'une main.



(a) Profondeur 6 (b) Profondeur 8 avec contrôle du temps

Figure 5.7 *Modélisation par octree temps réel à partir de* 5 *caméras.*

Une vidéo présente les résultats de cette modélisation temps réel en variant certains paramètres : présence ou non du contrôle du temps, profondeur maximale, etc. Cette vidéo est disponible à l'adresse suivante : http://perception.inrialpes.fr/people/Menier/Video/

5.5 Bilan

Nous avons présenté une parallélisation adaptative de la modélisation par octree basée sur un mécanisme de vol des tâches. L'utilisation d'un tel mécanisme permet de mieux équilibrer la charge de calcul de chaque processeur. Ceci est un facteur important pour l'efficacité de la parallélisation dans le cas de l'octree car les tâches sont de durées très variables. Une autre originalité mise en place dans cet algorithme consiste en la gestion d'un parcours en largeur. Ceci permet de contraindre le temps d'exécution du programme et d'obtenir à l'expiration du délai un niveau de détail homogène du modèle reconstruit. Les résultats expérimentaux permettent de valider l'efficacité de cette approche et de valider son intérêt dans un contexte interactif. La précision alors obtenue avec 16 processeurs et un délai de 30 ms avoisine le centimètre : cette précision est trois fois supérieure aux techniques volumiques distribuées précédentes.

Cette contribution n'est pas spécifique à la modélisation par octree et peut s'appliquer à tout parcours en largeur d'arbres et dont on peut arrêter l'exploration en approximant les résultats. Cette technique va ainsi être étudiée dans le contexte des simulations distribuées et en particulier pour la gestion des collisions. Ce travail fait l'objet d'une thèse réalisée par Everton Hermann.

Si la contrainte d'homogénéité globale du modèle est souvent judicieuse, il existe des

cas où l'application finale souhaite obtenir une meilleure précision sur une zone plutôt qu'une autre. Par exemple si le modèle doit servir à entrer en collision avec des objets, il est préférable de concentrer les calculs au voisinage des objets proches plutôt que dans des zones plus éloignées des objets. Pour prendre en compte de tels desirata, une amélioration consisterait à affecter une priorité à chaque noeud et à effectuer un parcours dans l'ordre de ces priorités.

III

Mise en œuvre

Implantation 6

La mise en oeuvre d'un système de vision multi-caméra présente certaines difficultés. Comme nous l'avons décrit, beaucoup de traitements différents sont mis en jeu nécessitant un très grand nombre d'échanges de données entre eux. Conserver la modularité et la flexibilité du concept du système dans sa mise en oeuvre présente aussi un enjeu. En particulier la généricité par rapport aux besoins des applications finales et par rapport au nombre de ressources (caméras ou machines) allouées à chaque étape est nécessaire pour permettre de déployer facilement ce système dans des contextes différents. Un grand soin doit aussi être apporté à l'effacité de cette implantation pour obtenir les meilleures performances possibles.

Pour faciliter la conception de tels systèmes modulaires, divers outils ont été développés dans des domaines tels que le parallélisme, la réalité virtuelle ou le génie logiciel. Chacun de ces outils présente des caractéristiques spécifiques aux applications visées. MPI [Mpi94], un des outils les plus utilisés en parallélisme, permet de développer des applications parallèles très efficaces. Son utilisation au début de la thèse nous a montré ses limites dans notre cas en ce qui concernait la modularité. L'utilisation du mécanisme des *communicateurs* est relativement lourde et nécessite un très gros développement : chaque communication entre étapes doit faire l'objet d'une déclaration au sein du programme lui-même. D'autres outils orientés composants comme des implantations parallèles [Denis 03] du standard CORBA [Corba95] permettent une grande flexibilité et généricité mais s'avèrent assez lents pour des applications effectuant beaucoup de communications. Nous avons choisi d'utiliser la bibliothèque FlowVR [Allard 04] développée par l'équipe MOAIS de l'INRIA. Cet intergiciel est dédié aux applications interactives distribuées et répond ainsi bien à nos besoins.

Nous présentons d'abord le modèle de conception de FlowVR pour mettre en avant

ses caractéristiques qui nous sont particulièrement utiles. Nous présentons ensuite l'implantation de notre système. Nous nous intéressons particulièrement aux méthodes que nous avons développées pour gérer de manière efficace les communications et les mécanismes de maintient de la cohérence dans le système asynchrone. Une partie de ce travail a été réalisée avec Jérémie Allard auteur de FlowVR. Nous détaillons enfin l'implantation des tâches d'acquisition et de traitement des images.

6.1 Le modèle FlowVR

FlowVR est un intergiciel de développement d'applications interactives modulaires. Cette restriction à ce type d'applications lui permet de proposer un modèle spécifique plus adapté et ainsi d'être plus efficace. La construction des applications repose sur une séparation entre la programmation de composants potentiellement parallélisés et la définition d'un graphe de flux de données contenant des mécanismes de filtrage et de synchronisation. Le modèle proposé comporte trois structures. Les *modules* représentent les composants effectuant à chaque itération un traitement. Les *messages* correspondent aux données transmises entre les modules. Enfin le *réseau abstrait* représente le graphe de flux de données permettant au système de transmettre les messages d'un module à un autre. Nous détaillons par la suite chacune de ces structures.

Les Modules

Un module correspond à un programme effectuant à chaque itération un traitement sur des données d'entrée pour fournir un ou plusieurs résultats en sortie. Ces données sont échangées avec le reste de l'application à travers une notion abstraite de ports d'entrée et ports de sortie. Chaque port est défini par un nom. Notons qu'à un traitement parallélisé peut correspondre soit un seul module (le parallélisme est masqué au reste de l'application) soit à plusieurs modules "identiques" répliqués (un module par processeur). L'exécution d'un module correspond généralement à une boucle ainsi décrite :

- attente du début de l'itération (opération *wait*) ;
- récupération des données d'entrée (opération *get*) ;
- calculs des résultats ;
- émission des résultats sur les ports correspondants (opération *put*).

Un point important est que chaque module n'a aucune connaissance – via FlowVR – des autres modules de l'application. Cela favorise la réutilisation d'un module dans diverses applications et simplifie la programmation de chaque module : il n'est pas nécessaire de tester la présence d'autres modules. La seule information fournie au module sur l'application provient du caractère *connecté* de ses ports. Un port est dit connecté si un tierce module émet ou reçoit des données de ce port. Ainsi un module peut déterminer la pertinence de calculer un résultat ou non en fonction de la connectivité du port servant à l'émission de ce résultat. De même la présence d'une entrée peut être détectée. Notons

que la connectivité est statique durant toute l'exécution de l'application.

Un module reçoit à chaque itération une et une seule donnée sur chacun de ses ports d'entrée connectés. Une seule donnée peut être émise par port de sortie et par itération. Le module peut cependant choisir de ne pas émettre un résultat à chaque itération.

Les messages

Les données véhiculées entre modules correspondent à des *messages*. Un message est un bloc de données associé à un certain nombre d'informations sémantiques appelées *estampilles*. Un message comporte certaines informations maintenues à jour automatiquement par FlowVR :

- le nom du module source ayant généré cette donnée ;
- le numéro d'itération du module source lors de l'envoi du message ;
- le numéro du message dans le flux de données. Ce numéro est incrémenté à chaque envoi d'un message sur un port. Il peut ainsi différer du numéro d'itération dans le cas où le module n'émet pas un message par itération.

En plus de ces estampilles prédéfinies, l'utilisateur peut ajouter d'autres informations utiles pour l'application (taille d'une image, numéro de la trame, etc.). Ces estampilles sont lisibles par chaque objet de l'application FlowVR et permettent d'effectuer des communications adaptées à l'application (filtrage selon une estampille sans avoir à analyser l'ensemble des données du message par exemple). Les estampilles sont définies par un nom et sont typées (*int*, *float*, *string*, *array*).

Le corps du message correspond en revanche à des données binaires brutes laissant aux applications la liberté du format. Ces données sont stockées dans un espace de mémoire partagée par les objets FlowVR. Ceci permet d'éviter les recopies lors de l'émission ou d'un échange local de messages (protocole zero-copy) et ainsi de gagner en performance.

Le réseau abstrait

Les ports d'entrée et de sortie des modules sont connectés entre eux via un réseau abstrait. Ce réseau forme le graphe de flux de données de l'application.

La primitive de base composant ce réseau est la *connexion*. Une connexion relie un port de sortie à un port d'entrée et transmet les messages en respectant l'ordre d'émission (mode FIFO). Un port d'entrée ne peut être relié qu'à un seul port de sortie. A l'inverse un port de sortie peut être connecté à plusieurs ports d'entrée.

Pour effectuer des traitements plus particuliers sur les messages, FlowVR apporte la notion de *filtres*. Comme les modules, les filtres communiquent avec le reste de l'application via des ports d'entrée et de sortie. Cependant les filtres diffèrent des modules selon plusieurs aspects :

- Un filtre n'est pas contraint à recevoir un et un seul message par port d'entrée. Il a accès à la liste complète des messages en attente sur chaque port. Il a la possibilité de sélectionner, combiner ou d'éliminer les messages. Il peut aussi créer et émettre

des nouveaux messages à n'importe quel moment et non au maximum un seul par itération comme pour les modules.
– Les filtres s'exécutent en tant que des extensions (*plugins*) des démons FlowVR et non comme des processus indépendants. Cela permet de gagner en performance évitant les changements de contexte lors des traitements des messages.

Les filtres permettent en particulier d'implanter des traitements légers sur les messages. FlowVR fournit par exemple des filtres de fusion (*gather*) ou de division (*scatter*) de messages, opérations très utiles pour la parallélisation des calculs. L'utilisation des filtres peut aussi être utile pour la compression ou la transformation de données ou encore la sélection des données les plus récentes pour rendre deux modules asynchrones comme nous le verrons par la suite.

FlowVR distingue une catégorie spéciale de filtres, appelés des *synchroniseurs*. Ces objets implantent les stratégies de couplage entre les différentes parties de l'application. Ils reçoivent les informations en provenance des filtres ou des modules concernés et émettent des "ordres" exécutés généralement par des filtres. Les synchroniseurs sont caractérisés par le fait qu'ils ne travaillent que sur les estampilles des messages et non sur leurs données. Ces informations sont donc très légères et peuvent ainsi être regroupées très rapidement pour prendre une décision centralisée.

Le réseau abstrait est ensuite instancié sur le réseau physique. FlowVR analyse les connexions et décide de l'envoi des messages sur le réseau physique lorsque cela est nécessaire. Il existe une différence fondamentale entre le réseau abstrait et les communications réseaux réellement effectuées. Des connexions dans le réseau abstrait peuvent être locales, c'est-à-dire concerner deux ports situés sur la même machine : aucun échange réseau n'est alors utilisé, FlowVR utilisant un mécanisme de passage de messages par mémoire partagée. A l'inverse le réseau abstrait peut comporter plusieurs connexions entre deux machines pour un même message. FlowVR détecte ces doublons et n'effectue qu'un seul échange réseau. De plus lors de l'instanciation de l'application il est possible de choisir le réseau physique à utiliser lorsque plusieurs sont disponibles.

La séparation entre la conception des modules et celle du réseau abstrait favorise la réutilisation des modules dans plusieurs applications. En pratique, les développeurs programment un ensemble de modules et peuvent les utiliser à leur guise dans diverses applications simplement en changeant le réseau abstrait.

Pour faciliter la conception du graphe du flux de données de l'application, FlowVR propose de décrire le réseau abstrait à travers un programme script. Grâce à des opérations haut niveau d'inter-connexion sur des groupes de modules, il est possible de concevoir des assemblages relativement complexes en quelques lignes. En particulier ces opérations permettent de gérer automatiquement la généricité par rapport aux degrés de parallélisation de chaque tâche. L'application est alors décrite par un fichier xml indiquant l'ensemble

des modules utilisés et un programme script de conception du réseau abstrait. Le changement d'un paramètre d'un module comme son degré de parallélisation nécessite ainsi que son changement dans le fichier xml, le script s'occupant automatiquement de regénérer le graphe du flux de données adéquat. Grâce à ces mécanismes de haut niveau il est possible de réaliser de très larges applications comportant plus de 10000 objets (connexions, filtres, modules). Un programme de lancement, fournit par FlowVR, permet alors en une seule commande de lancer l'application sur l'ensemble des machines. FlowVR fournit de plus divers outils très pratiques comme un visualisateur du réseau abstrait pour vérifier son exactitude. Nous avons aussi ajouté durant cette thèse un générateur de traces pour tenter d'analyser les performances de l'application.

6.2 Implantation

6.2.1 Implantation modulaire

Chaque tâche de la chaîne de traitement est implanté comme un module FlowVR : acquisition des images, soustraction de fond, filtres morphologiques sur les silhouettes, vectorisation de la silhouette, calcul des segments de vue et calcul de la surface. Remarquons que nous avons découpé la partie traitement d'images en une succesion de tâches chacune implantée dans un module séparé. Ceci permet de gagner en flexibilité : changer l'algorithme de soustraction de fond peut se faire sans changer la partie vectorisation des silhouettes et réciproquement. Grâce au mécanisme de transfert local sans recopie des messages via la mémoire partagée de FlowVR, implanter un traitement en plusieurs modules exécutés sur la même machine plutôt qu'en un seul n'a pas d'impact significatif sur les performances et permet de gagner en flexibilité.

Pour permettre aux modules de communiquer entre eux, il est important de définir le format des messages pour véhiculer les données. Pour cela nous avons développé une librairie d'encapsulation des données dans des messages FlowVR : images, silhouettes sous différents formats, maillage, etc... Cette librairie permet de masquer au développeur les détails de l'encapsulation. En particulier pour les images, les estampilles de taille, de format ou le numéro de trame sont automatiquement ajoutées aux messages.

6.2.2 Filtrage et schéma de communications collectives

L'efficacité de la distribution des tâches dépend principalement du coût des communications des données. Ces dernières peuvent être réduites en compressant les données à l'aide de filtres ou en utilisant des schémas de communications collectives lorsque plusieurs machines sont concernées.

Nous avons développé des filtres de compression des données en particulier pour les

images. Les silhouettes binaires peuvent être par exemple représentées selon plusieurs formats :

Matrice standard : chaque pixel est représenté par un octet dans un tableau bidimensionnel.

Matrice compacte : chaque pixel est représenté par un bit dans un tableau bidimensionnel. Cette représentation est particulièrement efficace pour les filtres morphologiques. Ce format est aussi 8 fois plus compact que la matrice standard. Il est en revanche plus difficile à manipuler, l'accés à un pixel individuel nécessitant plusieurs opérations pour extraire la valeur du bit.

RLE (Run-Length Encoding) : les successions de valeurs identiques sont codées par leur valeur et leur nombre. Pour une silhouette, ce mécanisme de compression est très efficace (environ 50 fois plus compact que la matrice standard). Ce format est donc particulièrement intéressant pour les transferts réseaux.

Les conversions entre ces formats sont extrêmement rapides prenant au maximum un demi millième de seconde pour la compression RLE. Il est donc rentable d'effectuer ces conversions pour réduire les temps de communication ou pour s'adapter aux besoins des traitements. Les images couleurs destinées à texturer le modèle peuvent aussi être grandement compressées en ne gardant que les pixels situés à l'intérieur de la silhouette. Ceci permet de réduire la taille de ces textures d'un tiers ou d'un quart.

D'autres filtres ont aussi été développés en particulier pour la gestion des flux de données : un mécanisme de *tourniquet* a été par exemple créé pour permettre de distribuer les trames à différentes unités de traitement.

Une autre amélioration pour réduire les coûts de communication consiste à utiliser des schémas de communications collectives lorsqu'une communication concerne simultanément plusieurs machines. FlowVR fournit des schémas par défaut reposant sur une structure d'arbres pour le *Broadcast* (diffusion d'un message à plusieurs modules), le *Scatter* (répartition des données contenues dans un message sur un ensemble de modules) ou le *Gather* (fusion des données en provenance de plusieurs modules). D'autres schémas s'avèrent plus efficaces pour des messages très volumineux ou pour des opérations combinées comme un *All-Gather* (*Gather-Broadcast*) en découpant les messages en morceaux plus petits et en utilisant des techniques similaires à celles utilisées dans les réseaux pair à pair [Barnett 94]. Ces schémas avancés sont particulièrement intéressants lorsque le degré de parallélisation est grand ou que les données sont très volumineuses comme par exemple dans le cas des textures.

6.2.3 Gestion de l'asynchronisme

Les communications sur les connexions de base étant en mode FIFO, tous les composants de l'application s'exécutent de manière synchrone. Ceci limite le débit à la fréquence du traitement le plus lent. Lorsque la scène n'est pas trop complexe le système est limité par la fréquence d'acquisition des caméras. En revanche lorsque la scène devient plus complexe, par exemple lorsque plusieurs personnes ou objets sont introduits, certains traitements comme la modélisation des formes deviennent beaucoup plus coûteux et ne peuvent traiter les trames à la fréquence de l'acquisition. Les trames s'accumulent alors à l'entrée de ces modules provoquant une augmentation très rapide de la latence, jusqu'à arriver à la saturation des tampons. Pour éviter cela il est nécessaire de désynchroniser les traitements pour que le système puisse conserver des performances optimales même lorsque la scène devient très complexe. Le principe est d'utiliser pour les traitements coûteux un mécanisme "best-effort" de sous-échantillonage leur permettant de travailler avec une fréquence moindre. Ce mécanisme repose sur l'utilisation d'un synchroniseur glouton (*Greedy*) qui permet à chaque nouvelle itération de travailler sur les derniers messages arrivés. Ainsi le traitement ne prend aucun retard et s'exécute à sa fréquence maximale. Remarquons que ce mécanisme fournit les même performances que le mode FIFO lorsque les traitements peuvent s'exécuter à la fréquence d'acquisition des caméras. En particulier, contrairement au mécanisme classique du glouton, nous n'autorisons pas les modules à s'éxecuter à des débits supérieurs car alors ils traiteraient inutilement plusieurs fois la même trame.

Cet asynchronisme entraîne deux difficultés. La première difficulté est liée au maintient de la cohérence dans l'application. Si un module sélectionne indépendamment les messages les plus récents sur chaque port, comme le proposent Hasenfratz *et al.* [Hasenfratz 04], ces données d'entrée à chaque itération ne seront pas garanties de correspondre à une même trame. Pour garantir cette cohérence des données, il est important lors de la sélection des messages de vérifier leur appartenance à une même trame. Pour cela nous utilisons un synchroniseur centralisé se basant sur les estampilles indiquant le numéro de la trame correspondant à chaque message. Lorsque le module ne peut traiter suffisamment vite les informations, ce synchroniseur décide de jeter **tous** les messages correspondant à une trame, pour passer directement à la suivante. La cohérence des messages est ainsi garantie.

La seconde difficulté concerne le choix de la politique d'asynchronisme en fonction des critères de performance. Dans le paragraphe précédent nous avons présenté une stratégie simple consistant à sélectionner à chaque itération la trame la plus récente. Cette stratégie offre une garantie du meilleur débit étant donné que le programme n'attend jamais une donnée plus longtemps que dans le mode FIFO. En revanche cette technique n'est pas optimale en ce qui concerne la latence comme le montre la figure 6.1. Une

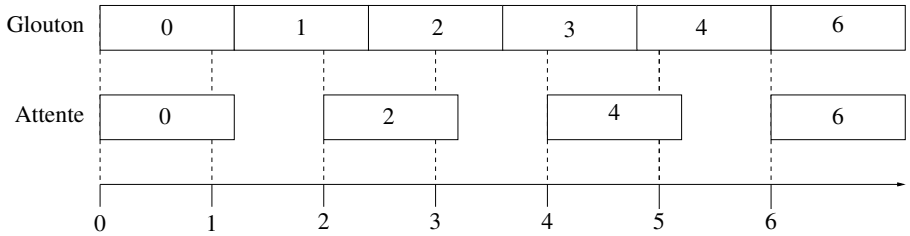


Figure 6.1 *Deux politiques d'asynchronisme différentes sur une entrée à* 1 *Hz et un temps de calcul légérement supérieur à une seconde. La politique glouton utilise le dernier message disponible : le débit est optimal mais pas la latence. La seconde politique attend l'arrivée d'un nouveau message. La latence est alors optimale mais plus le débit.*

autre solution optimale pour la latence mais pas pour le débit consiste à chaque itération à attendre la prochaine trame (voir figure 6.1). Le choix de la stratégie d'asynchronisme influe donc sur les performances du système et il est important pour chaque application de sélectionner la plus adaptée. Une autre façon de voir le problème consiste à faire une analogie avec une chaîne de cuisson MacDonald. Le débit correspond à la fréquence de distribution des sandwichs aux clients et la latence correspond à la fraîcheur du produit (délai entre sa cuisson et sa distribution). Plus l'espace de stockage des sandwichs cuits est grand, meilleur est le débit, les variations des temps de cuisson étant alors amortis par le stockage. En revanche moins bonne sera la fraîcheur des sandwichs ceux-ci ayant attendu avant d'être servi. Dans cette analogie le comportement glouton précédemment décrit correspond à ne stocker qu'un seul sandwich. Pour obtenir une fraîcheur idéale le mécanisme d'attente ne fournit aucun espace de stockage et oblige les clients à attendre qu'un sandwich arrive. Le choix de la dimension de stockage influe donc sur les performances du système. D'autres politiques plus complexes peuvent aussi être mises en place comme celle utilisée réellement par MacDonald : l'espace de stockage est très grand, mais un sandwich est jeté si il est resté plus d'un certain laps de temps en attente. Ainsi un vaste de choix de politiques d'asynchronisme est disponible chacune ayant ses spécificités.

6.3 Acquisition et traitements des images

Nous détaillons ici les modules que nous avons développés pour l'acquisition et les traitements des images. Ces techniques proviennent d'une synthèse des méthodes proposées dans la littérature et le cas échéant de leur modification pour mieux correspondre à nos besoins. Le choix de ces méthodes prend en compte la qualité de leurs résultats mais aussi leurs performances et leur adéquation avec les besoins des modélisations.

6.3.1 Acquisition synchrone des images

L'acquisition s'effectue sur un ensemble de PCs dédiés à cette tâche, chacun gérant une seule caméra. Nous utilisons des caméras numériques standard FireWire (IEEE 1394) avec une résolution de 780×580. Elles proposent nativement différents formats d'image en particulier le format YUV qui s'avère très pratique pour l'étape suivante d'extraction de silhouettes. L'acquisition de ces images est réalisée grâce à la librairie Blinky. Etant donné que nous utilisons plusieurs capteurs, il est nécessaire de s'intéresser au problème de la synchronisation entre eux. Il est en effet important que les informations issues des différentes caméras soient cohérentes, c'est-à-dire soient relatives à un même événement. Nous utilisons un mécanisme de synchronisation électronique développé en interne qui déclenche simultanément l'ensemble des caméras. Ainsi les images obtenues comportent un délai entre elles de moins de $100\mu s$. Notons que notre système peut fonctionner sans mécanisme de synchronisation externe : le délai entre les images peut alors être de l'ordre du délai entre deux trames (33 ms pour une acquisition à 30 images par seconde). La précision des informations 3D extraites est alors fortement diminuée les images n'étant plus parfaitement cohérentes. Ce phénomène est particulièrement gênant lors de mouvements rapides (ex : les mains peuvent avoir bougées de plusieurs centimètres entre deux images).

Le réglage et le calibrage de ces caméras sont assurés par une suite logicielle développée au sein de l'équipe PERCEPTION.

6.3.2 Extraction des silhouettes

Les régions d'intérêt des images sont identifiées pour ne retenir que la partie correspondant à l'utilisateur : la silhouette. Une solution de soustraction de fond couremment utilisée dans l'audiovisuel est le chroma-keying. Un fond de couleur mate (bleu ou vert) est disposé tout autour de la scène derrière l'utilisateur. Pour chaque pixel on estime la distance à cette couleur de référence pour décider si il est issu du fond ou de l'utilisateur. Les silhouettes ainsi obtenues sont pratiquement parfaites. Cependant cette technique requiert un environnement très contrôlé, en particulier au niveau des éclairages et des habits autorisés pour l'utilisateur. D'autres solutions ne faisant pas l'hypothèse d'un fond uniforme ont été proposées dans la littérature [Horprasert 99, Cheung 00, Shen 04]. La vaste majorité d'entre-elles repose sur l'hypothèse d'un fond fixe dont un modèle statistique est appris au préalable. Chaque image acquise est alors comparée à ce modèle pixel par pixel : si la couleur du pixel correspond au modèle, celui-ci est classifié en fond, sinon il est considéré comme appartenant à la silhouette. La difficulté principale de ces techniques est la gestion des changements d'éclairage, en particulier des ombres. Horpraesert *et al.* [Horprasert 99] proposent de séparer les traitements de la luminosité et de la chrominance pour détecter ces changements d'illumination. Après de nombreux tests, nous avons donc choisi d'utiliser une variante de cette méthode en modélisant la chrominance (UV) par une distribution gaussienne corrélée et l'intensité lumineuse (Y) par un modèle intervalle. Les

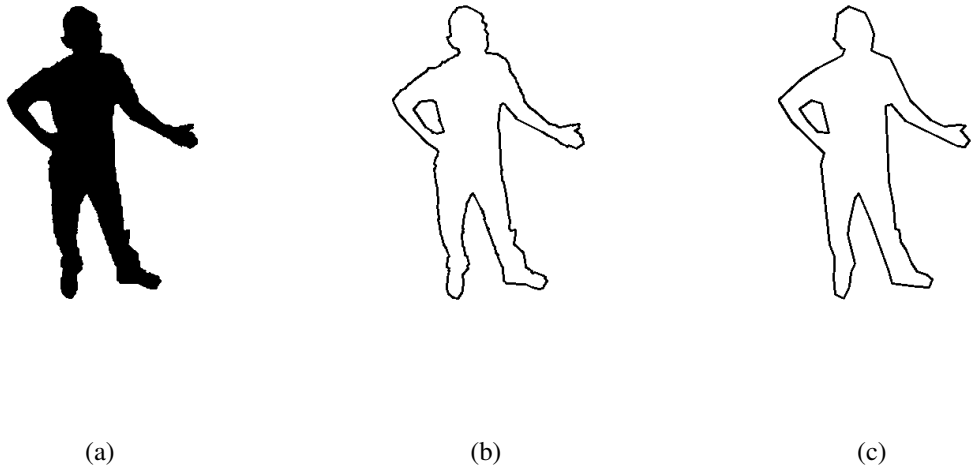


Figure 6.2 *Vectorisation de la silhouette. (a) Silhouette binaire d'entrée. (b) Vectorisation exacte (*250 *sommets). (c) Vectorisation approximée avec une tolérance de* 3 *pixels (*55 *sommets).*

images binaires obtenues sont ensuite filtrées par des opérateurs morphologiques d'érosion et de dilatation pour éliminer une partie du bruit résiduel. Cette méthode malgré sa simplicité fournit de bons résultats et permet une exécution rapide (cf section 7.3 page 82). Pour une étude comparative des différents algorithmes de soustraction de fond statique, nous renvoyons le lecteur au travail de Karaman *et al.* [Karaman 05]. D'autres techniques ne font pas l'hypothèse d'un fond statique et permettent un apprentissage progressif à la volée du fond. Ces techniques sont cependant difficilement utilisables dans un système d'interaction : l'utilisateur risque d'être considéré comme appartenant au fond si une partie de son corps (les jambes par exemple) reste immobile trop longtemps.

6.3.3 Vectorisation des silhouettes

Pour la vectorisation des contours, Franco *et al.* [Franco 05] proposent d'utiliser une méthode développée par Debled *et al.* [DebledRennesson 95]. Cet algorithme permet d'obtenir une vectorisation exacte dans le sens où le rendu, par l'algorithme de Bresenham, du polygône résultant est exactement le contour initial. De plus le nombre de segments est relativement faible (environ 200 pour une personne) dans le cas de silhouettes parfaites. En revanche cet algorithme est sensible au bruit et le nombre de segments augmente très fortement lors de la présence de bruits dans les contours. Pour palier à cela nous proposons d'utiliser une variante de cet algorithme proposée

récemment par Debled *et al.* [DebledRennesson 04] qui permet une vectorisation dégradée avec contrôle de l'erreur maximale par rapport à la vectorisation exacte. Ceci permet de fortement réduire le nombre de segments même pour des niveaux de tolérance très faible (2 ou 3 pixels). La figure 6.2 présente des résultats de vectorisation avec ces algorithmes.

Remarquons qu'il est important lors de cette vectorisation de bien tenir compte de l'orientation des contours comme décrit dans les travaux de Franco [Franco 05].

6.4 Bilan

Ce chapitre a présenté les détails de l'implantation de notre système d'interaction multi-caméra. Celle-ci repose sur l'utilisation de la librairie FlowVR. Son caractère dédié aux applications interactives distribuées lui permet de fournir un modèle de programmation simplifié conduisant à de meilleures performances que les librairies génériques comme MPI ou les implantations de Corba. FlowVR repose sur une séparation entre les composants de calculs, les modules qui consistent en une boucle itérative, et le réseau abstrait en charge de faire véhiculer les messages entre les différents modules. Cette séparation permet d'obtenir une grande flexibilité dans la conception du système. Nous avons créé un ensemble de scripts permettant de générer le réseau abstrait en fonction du nombre de machines ou de caméras et des besoins des applications finales.

Pour obtenir les meilleures performances possibles en terme de débit mais surtout de latence, nous avons proposé plusieurs améliorations principalement sur le transfert des données : compression adaptée des données, asynchronisme des modules tout en préservant la cohérence des données.

Enfin nous avons fourni les détails des implantations des modules en charge de l'acquisition et des traitements d'image. La plupart repose sur une synthèse des approches existantes comme la soustraction de fond. Une contribution importante consiste en l'utilisation d'un algorithme de vectorisation dégradée permettant de contrôler la précision des données et ainsi d'influer sur les performances du système. En particulier, il est fréquent qu'avec des tolérances assez faibles, un gain significatif de latence soit obtenu sans pour autant diminuer de façon significative la précision des modèles reconstruits.

Plusieurs améliorations sont encore envisageables. La plus importante, et certainement celle qui aurait un impact le plus fort sur la qualité des données extraites, serait l'amélioration de la soustraction de fond. En effet dans les situations réelles il est fréquent que des défauts apparaissent lorsqu'un utilisateur porte un vêtement de couleur trop proche de celle du fond. L'utilisation de méthodes plus coûteuses mais plus performantes,

comme la prise en compte de l'information de texture (gradient) ou de profondeur (stéréo), permettrait d'obtenir des silhouettes plus fiables. Le coût supplémentaire que cela engendrerait peut être désormais facilement réduit grâce aux capacités de programmation accrues des nouvelles cartes graphiques comme le permet la série G80 de NVIDIA. Un gain significatif peut aussi être obtenu en utilisant un mécanisme de *graph-cut* pour passer des probabilités à l'image binaire : ceci permet en particulier de régulariser les silhouettes et ainsi d'éviter les petites erreurs. Si cette technique n'est pas envisageable avec un seul processeur (elle nécessite 100 ms par image), il est possible de découper l'image en plusieurs zones, chacune traitée par un processeur distinct : le résultat n'est alors pas exact mais pourrait largement suffir.

Applications 7

Pour valider notre approche, nous avons implanté notre système pour différentes applications interactives. Ces applications mettent en avant les différentes modélisations des formes et démontrent l'utilité de notre système multi-caméra pour mesurer des informations sur l'utilisateur et s'en servir pour des interactions. Ces applications ne sont bien sûr que des exemples des possibilités du système.

7.1 La plateforme GrImage

Les applications présentées dans cette thèse ont été développées pour la plupart sur la plateforme GrImage [Allard 06] ou sur sa version portable baptisée mini-Grimage. L'objectif de cette plateforme est de fédérer les travaux de plusieurs équipes de recherche en vision par ordinateur, parallélisme, graphisme et animation autour de projets communs. En particulier les applications interactives que nous montrons dans ce chapitre sont le fruit de collaborations avec les équipes EVASION et MOAIS.

GrImage est une plateforme regroupant plusieurs caméras (environ une dizaine d'AVT Marlin F-043C) et un mur de projection de 16 vidéo-projecteurs permettant d'immerger l'utilisateur dans un univers virtuel de haute qualité (12 millions de pixels sur un écran de 3×2 m). Une grappe de PC permet de piloter l'ensemble de ces ressources et d'effectuer les calculs. Elle est composée de 11 bi-Xeon 2.66 GHz et de 16 bi-Opteron 2.0 GHz reliés entre eux par un double réseau Ethernet Gigabit. Une machine à mémoire partagée comportant 8 dual-core Opteron 2.2 GHz est aussi disponible via une liaison 10 Gigabits fibre optique. Dans la suite les 16 bi-Opteron sont utilisés pour la visualisation ou les

simulations interactives, la partie vision par ordinateur du système s'exécutant sur les bi-Xeon.

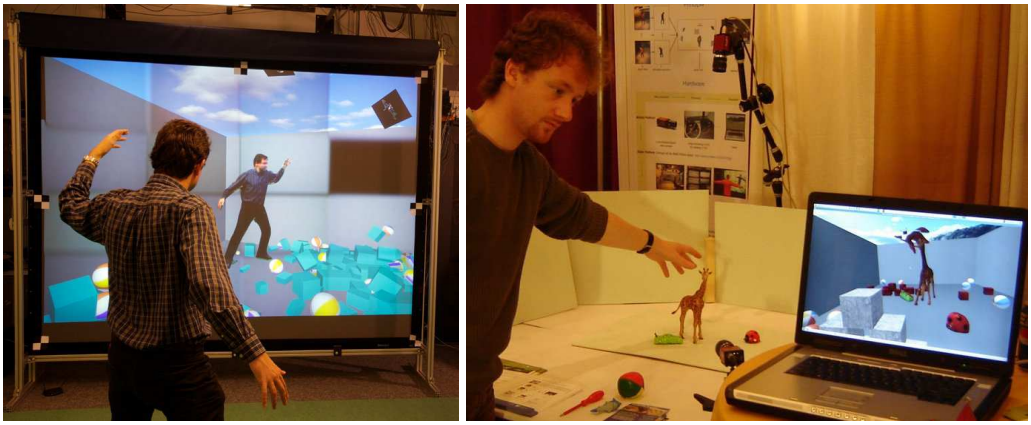
La version transportable de cette plateforme, mini-Grimage, comporte seulement 4 caméras AVT Marlin 640×480, 6 mini-PC Pentium-4 3.2 GHz et un ordinateur portable pour effectuer la visualisation et les simulations.

Nous pouvons remarquer que ces plateformes ne sont composées que de matériel relativement standard et peuvent donc facilement être modifiées (ce qui fut le cas durant la thèse). Notre système, de par sa flexibilité, peut s'adapter à n'importe quelle plateforme et n'est pas dédiée à GrImage. En particulier il a pu être testé sur une plateforme de l'université de Brown utilisant une grappe IBM à base de bi-Pentium et des caméras IP de surveillance fournissant des flux JPEG d'images non synchronisés.

7.2 Applications interactives

Pour démontrer la générécité et la flexibilité de notre système nous avons réalisé différentes applications interactives mettant en oeuvre les modélisations des formes distribuées présentées.

7.2.1 Intégration visuelle



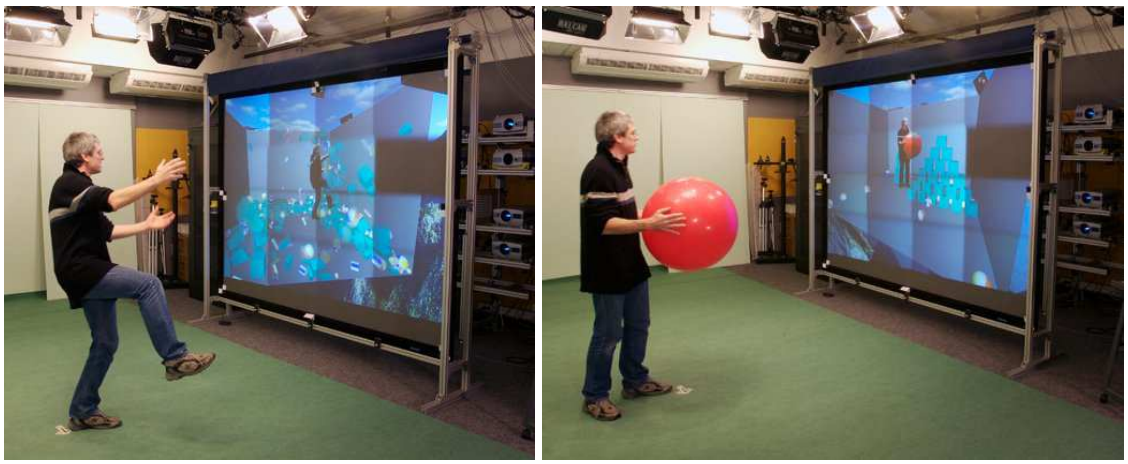
(a) (b)

Figure 7.1 *Modélisation $3D$ de l'utilisateur en temps réel et intégration de l'utilisateur dans le monde virtuel.*

La première interaction consiste simplement en un retour visuel : l'utilisateur est intégré au sein de l'univers virtuel. Grâce au modèle géométrique reconstruit de la surface de l'utilisateur en temps réel, il est possible de générer l'image de ce dernier selon un

nouveau point de vue même éloigné de ceux des caméras. L'information photométrique est ajoutée grâce à la projection des images couleur sur le modèle (voir figure 7.1) en utilisant la technique décrite dans la section 2.2.2 page 17. Cet ajout des textures apporte énormément pour le réalisme de l'intégration. La difficulté de ce sytème est de bien coordonner le flux des modèles et les textures issues de chaque caméra. En particulier il est important de maintenir une certaine cohérence, malgré l'asynchronisme, pour qu'un modèle géométrique soit rendu avec les textures de la trame lui correspondant. Ces informations de forme et de photométrie n'ont pas la même latence : il est nécessaire d'attendre la disponibilité de l'une avant d'envoyer l'autre à la carte graphique. Une vidéo présentant cette intégration de l'utilisateur dans un environnement virtuel est disponible : `http://perception.inrialpes.fr/people/Menier/video/`.

7.2.2 Interactions avec des objets rigides



(a) (b)

Figure 7.2 *Interactions entre l'utilisateur et des objets rigides virtuels : les collisions entre le modèle géométrique et les objets sont calculés par le moteur ODE.*

Une autre application utilisant l'information de forme sur l'utilisateur consiste en l'interaction avec des objets rigides virtuels. Nous utilisons pour cela le moteur de simulation physique ODE (*Open Dynamics Engine*) qui permet de gérer les collisions entre notre modèle surfacique et les objets virtuels (voir figure 7.2). La modélisation exacte de l'enveloppe visuelle prend alors toute son importance car il est important pour cette application de fournir un maillage triangulé fermé et sans auto-intersection. L'asynchronisme est aussi très important en particulier entre la modélisation et la simulation, cette dernière devant fonctionner à au moins 50 Hz pour la stabilité des collisions.

Cette application permet de montrer que la latence de la modélisation est suffisemment faible (entre 60 ms et 80 ms) pour être utilisée en remplacement des capteurs standards. Bien que cette latence soit constatable, les utilisateurs tendent à la corriger inconsciemment au bout de quelques secondes d'utilisation : ainsi ellle ne gêne pas trop l'interaction. Ces interactions ont été démontrées lors de la conférence IEEE VR'06 (vidéo disponible : `http://perception.inrialpes.fr/people/Menier/video`), du salon professionnel VISION 2006 ou lors de journées portes ouvertes durant la Fête de la Science. Ces démonstrations ont fonctionné pendant plus d'une centaine d'heures démontrant ainsi la robustesse de notre système même dans les situations très complexes (lors d'une démonstration une vingtaine de personnes se sont amusées à rentrer dans l'espace d'acquisition). De plus, le système ne nécessitant pas d'équipements spéciaux, n'importe qui peut interagir avec le monde virtuel sans contrainte : le système apparait comme totalement intuitif aussi bien pour les enfants que pour les adultes. En particulier les utilisateurs peuvent ammener des objets particuliers qui sont automatiquement reconstruits (voir figure 7.2(b)).

Une limitation à cette interaction est rapidement apparue lorsque les utilisateurs cherchent à donner un coup de pied dans un objet. Ce dernier est alors simplement poussé mollement, la vitesse lors de la collision n'étant pas prise en compte. L'extraction de cette information de vitesse permettrait de résoudre sans doute cette limitation. Une discussion sur l'utilisation de telles informations est présentée dans la conclusion générale à ces travaux de thèse au chapitre 11 page 115.

7.2.3 Perruque virtuelle

Figure 7.3 *Perruque virtuelle : interactions entre l'utilisateur et une simulation de cheveux grâce à l'ajout de l'information d'identification de la tête.*

Nous avons aussi utilisé les informations de positionnement de l'utilisateur pour suivre la position de la tête. En reliant cette dernière à une simulation de cheveux, une perruque virtuelle a pu être rajoutée sur l'utilisateur (voir figure 7.3). Cette information d'identification est extraite de façon ad-hoc très simplement : le point le plus haut du modèle est utilisé pour fixer une zone dans laquelle on calcule un centre de masse grâce à une reconstruction voxelique. La simulation de cheveux a été développée par Florence Bertails [Bertails 06] dans le cadre de sa thèse.

Cette application a mis en lumière les difficultés posées par le couplage multi-fréquence. La position de la tête n'est mesurée qu'à 30 trames par seconde alors que la simulation de la chevelure fonctionne à plus de 100 Hz. Avec un échantillonage naïf consistant à donner la position de la tête la plus récente, la position en entrée de la simulation est constante pendant quelques itérations puis "saute" à la position suivante. Ces à-coups rendent la simulation instable. Pour éviter cela nous avons utilisé des filtres d'interpolation pour lisser ces mouvements.

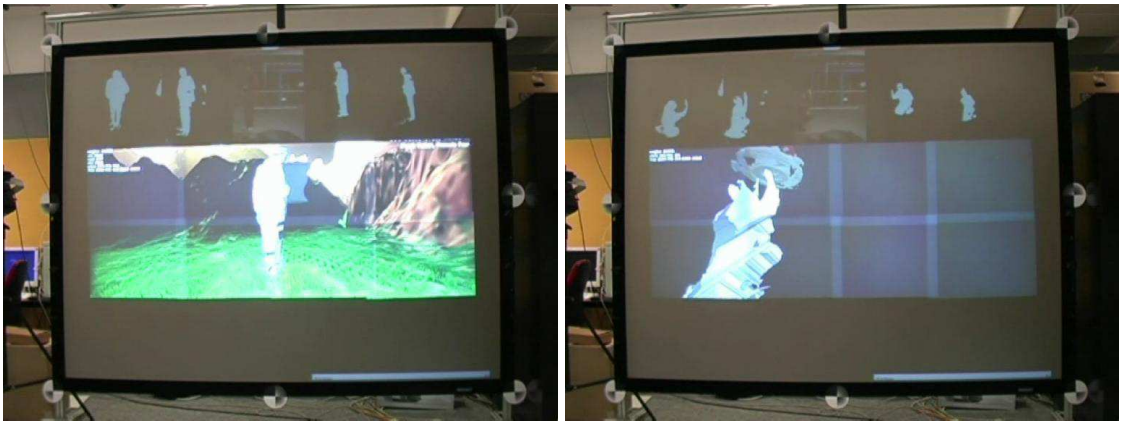
Cet exemple nous fournit une première limite de notre système en ce qui concerne les problèmes liès à la relative basse fréquence de l'acquisition des modèles comparée aux simulations. Cette limitation peut être résolue par une interpolation des données lorsque cela est possible. Ce type de mécanisme, très utilisée dans la communauté de la réalité virtuelle [Singhal 00, Margery 02], peut être facilement mis en place par des filtres FlowVR. Cette application permet en outre de démontrer le fort potentiel de l'ajout d'une information d'identification aussi basique soit elle.

Un point particulièrement intéressant avec cette démonstration réside dans le fait que les calculs de modélisation ont été réalisés par des serveurs situés à plusieurs kilomètres des caméras. Ainsi les informations de silhouettes et de forme étaient véhiculées par le réseau très haut débit VTHD. Cela démontre la possibilité d'effectuer des calculs déportés lorsque la puissance de calcul localement disponible n'est pas suffisante.

Notons que cette collaboration avec Florence Bertails fut prolongée par un travail annexe à cette thèse sur le rendu parallélisé de la chevelure [Bertails 05].

7.2.4 Sculpture

La modélisation volumique sous forme d'octree a été utilisée pour réaliser une sculpture virtuelle. Selon le mode d'interaction il est possible de supprimer de la matière, d'en ajouter ou d'en changer la couleur. Cette application met en avant les avantages qu'offre une modélisation complète du corps : modeler une forme de vase est par exemple très facile en utilisant son coude (voir figure 7.4). Une vidéo présentant cette application est disponible : `http://perception.inrialpes.fr/people/Menier/video`



(a) (b)

Figure 7.4 *Sculpture virtuelle grâce à la modélisation volumique sous forme d'octree : l'utilisateur peut ajouter, supprimer de la matière ou changer sa couleur.*

La principale difficulté rencontrée lors de l'utilisation de cette application concerne le rendu non-immersif : l'utilisateur a du mal à se positionner précisément par rapport à l'objet virtuel. Cette difficulté était d'autant plus aggravée que la scène était visualisée avec un écran $2D$ et non avec un mode stéréoscopique $3D$.

7.2.5 Interactions avec un fluide

La dernière application développée consiste en l'interaction avec une simulation de fluide eulérien. Cette simulation est basée sur une grille régulière de $32 \times 32 \times 64$ cellules. Nous avons donc utilisé une modélisation volumique pour réaliser les interactions entre l'utilisateur et le fluide. Lors de cette application la modélisation était séquentielle étant donné que la précision est limitée par celle de la grille du fluide. Pour des fluides plus précis nous aurions utilisé une modélisation distribuée comme détaillée au chapitre 5 page 49.

Nous avons aussi couplé cette interaction avec la simulation des objets rigides présentée précédemment (voir figure 7.5). Ceci implique de mettre en jeu simultanément les modélisations volumiques et surfaciques. Une vidéo de cette application est disponible : `http://perception.inrialpes.fr/people/Menier/video`

7.3 Performances

Lors de ces applications nous avons étudié les performances obtenues par notre système aussi bien sur le plan des temps de calcul avec le débit et la latence qu'en terme de

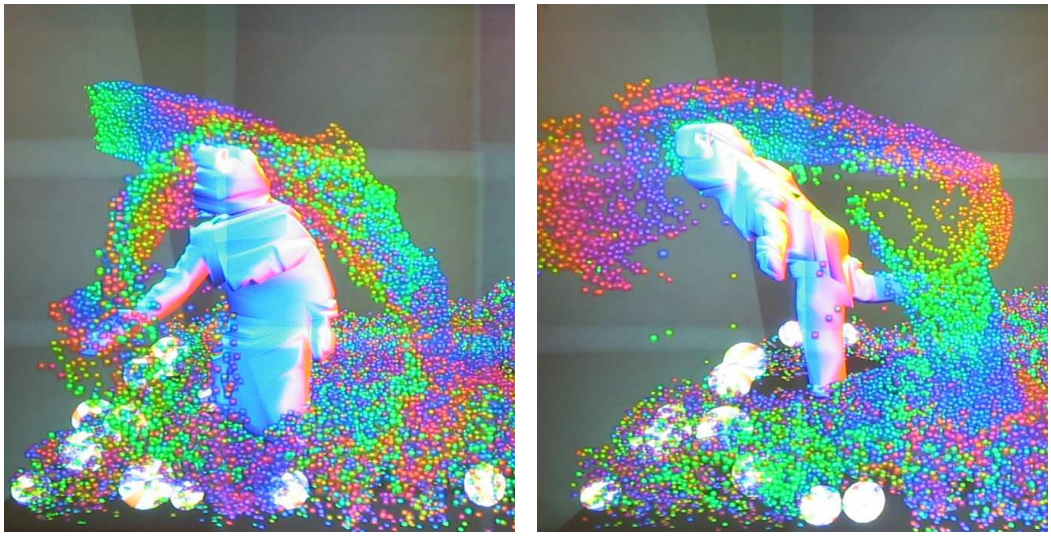


Figure 7.5 *Interactions avec un fluide et couplage avec l'interaction basée sur la simulation physique des objets rigides : les modélisations volumiques et surfaciques sont alors mises en jeu.*

précision de la reconstruction.

7.3.1 Débit et latence

Nous avons mesuré les performances de notre système lors d'une utilisation classique sur la plateforme GrImage, c'est-à-dire où un seul utilisateur est filmé depuis 6 points de vue. Chaque caméra est gérée par une machine qui lui est dédiée et 4 processeurs sont dédiés à la modélisation surfacique (2 CPUs pour le calcul des segments de vue et 2 CPUs pour l'extraction de la surface). Le tableau 7.1 résume les temps moyens pour traiter une trame par chaque étape du système. La latence du système est alors d'environ 70 ms. En ce qui concerne le débit, le système permet sans souci de tenir le débit standard de 30 trames par seconde comme le montre le schéma 7.6. En augmentant la fréquence d'acquisition des caméras, nous avons pu valider notre système jusqu'à un débit de 40 trames par seconde. Nous n'avons en revanche pas pu tester à des débits supérieurs, les caméras ne supportant pas une acquisition plus rapide.

7.3.2 Précision

Un autre facteur important dans notre système est la précision des modèles obtenus. La modélisation surfacique étant exacte par rapport aux silhouettes vectorisées, la qualité dépend donc de la précision de la soustraction de fond et de la vectorisation des silhouettes.

Étape	temps
Acquisition	1 ms
Soustraction de fond	15 ms
Filtres morphologiques	1 ms
Vectorisation de la silhouette	1 ms
Calcul des segments de vue	20 ms
Extraction de la surface	25 ms
Total des communications	$\approx$ 7 ms
Latence	$\approx$ 70 ms

TAB. 7.1 *Temps moyens d'exécution de chaque étape pour* 6 *caméras* 780×582 *sur des Xeon* 2.66 *GHz avec* 2 *processeurs pour chaque étape de la modélisation.*

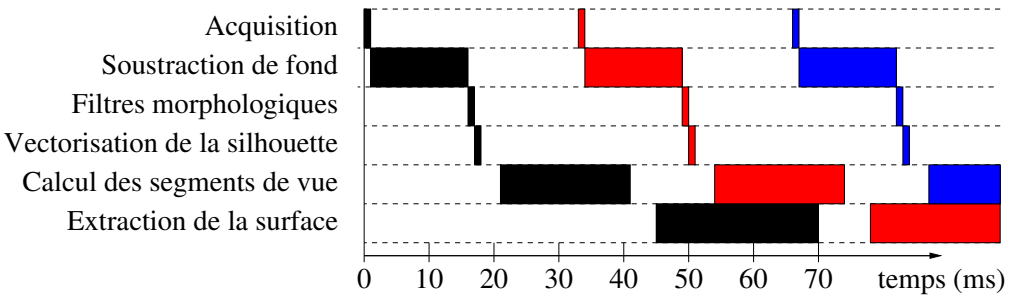


Figure 7.6 *Activité du pipeline : les couleurs représentent les différentes trames.*

Dans le cas idéal où la soustraction de fond est parfaite et en utilisant l'algorithme de vectorisation exact, la précision du modèle correspond à la taille d'un pixel, environ un tiers de centimètre dans notre cas. Dans la pratique la soustraction de fond n'est pas parfaite et nous utilisons une vectorisation avec une tolérance d'un pixel d'erreur. Malgré cela, les modèles sont relativement précis et une précision d'environ un centimètre peut tout de même être atteinte. Il est ainsi possible d'obtenir les doigts de la personne malgré une modélisation complète d'un espace de $2m \times 2m \times 2m$ comme le montre la figure 7.7.

7.4 Bilan

Nous avons montré dans ce chapitre plusieurs applications réalisées à l'aide de notre système d'interaction. Ces expériences démontrent, si cela était encore nécessaire, l'intérêt des systèmes multi-caméras temps réel et de l'acquisition de la géométrie complète de l'utilisateur pour les applications interactives $3D$. Grâce à ces tests plusieurs aspects ont pu être validés.

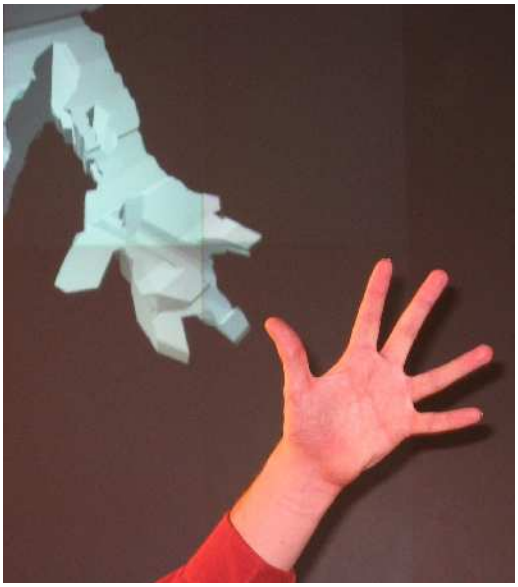


Figure 7.7 *Précision de la modélisation des formes : zoom sur la main lors de la modélisation complète de l'utilisateur.*

La diversité des applications proposées permet de conclure que notre système est capable de s'adapter aux différents besoins d'un grand nombre d'applications. Ceci est principalement lié au travail de formalisme réalisé sur l'ensemble du système et à l'utilisation d'une approche modulaire. Un point très important, que démontre la dernière interaction présentée, est la nécessité dans certaines situations d'effectuer plusieurs modélisations des formes en parallèle, chacune ayant sa spécificité. Ceci est rarement pris en compte dans les systèmes de vision classiquement proposés qui cherchent à utiliser une unique modélisation pour l'ensemble de leur application.

Malgré les problèmes de soustraction de fond dans les conditions réelles, la précision obtenue d'environ un centimètre est très satisfaisante et ne peut qu'augmenter avec l'amélioration de ce traitement critique et avec l'augmentation des résolutions des caméras.

Grâce au parallélisme, l'obtention de modèles relativement précis est réalisée avec des performances très satisfaisantes. Le débit avoisine la fréquence d'acquisition des caméras vidéo standards, 30 images par seconde, et la latence mesurée est d'environ 70 ms. La latence perçue est en réalité un peu plus grande : ceci est du au temps d'acquisition par la caméra et au temps d'affichage par la carte graphique et les vidéo-projecteurs. Les travaux de Stanislaw Borkowski [Borkowski 04] ont permis d'établir que la latence est alors augmentée de 30 ms. Nous obtenons donc une latence totale d'environ 100 ms, ce qui reste relativement bon.

Notre système d'interaction répond donc correctement aux critères que nous souhaitions résoudre au départ : obtenir des modèles complets de bonne précision, en temps réel et avec une faible latence.

Ces expériences ont aussi pu valider le fait que notre approche pouvait fonctionner sur d'autres plateformes que GrImage. Notre système s'adapte très simplement aux différentes configurations matérielles qu'elles soient plus restreintes comme sur la plateforme mini-GrImage ou qu'elles nécessitent des aménagements plus particuliers comme lors de la Fête de la Science où la modélisation était effectuée sur des machines situées à plus d'une dizaine de kilomètres des caméras et de l'affichage. Cette flexibilité par rapport au matériel est très importante car elle permet de déployer notre système dans un maximum de situations. Ces expériences valident ainsi l'intérêt de l'utilisation d'un logiciel orienté composants tel que FlowVR pour le développement et le déploiement de telles applications. De plus ces applications ont permis de tester notre système pendant plusieurs centaines d'heures démontrant ainsi sa robustesse malgré des utilisations parfois extrêmes avec une dizaine de personnes au centre de l'espace d'acquisition par exemple.

Toutes ces applications ne sont cependant que des démonstrations (applications jouets) permettant de valider notre approche. Il serait intéressant d'étudier notre système d'interaction avec la vision de la communauté des interfaces Homme-Machine. Ainsi nous nous sommes rendus compte que certaines interactions apparaissaient très naturelles alors que d'autres beaucoup moins. Ces idées seront développées dans les perspectives générales de la thèse données dans la partie V page 113.

IV

Modélisation du mouvement

État de l'art 8

Nous avons présenté dans les parties précédentes la modélisation de la forme de l'utilisateur et son utilisation dans des applications interactives. Cette modélisation fournit pour chaque trame un maillage indépendant : aucun lien n'est fourni entre les données des maillages issus de trames distinctes. De plus aucune signification n'est apportée sur ce que représentent les facettes ou les points. Ces informations de suivi et d'identification peuvent s'avérer très utiles pour certaines applications interactives. En particulier celles-ci permettent de calculer la vitesse et l'accélération en chaque point du maillage : ces données permettent d'améliorer grandement le réalisme des collisions en autorisant l'utilisateur à transférer de l'énergie cinétique aux objets. Ces informations sont aussi vitales pour l'utilisation du maillage dans des simulations implicites où des dérivées temporelles doivent être calculées.

Une approche répondant simultanément au problème du suivi temporel et de l'identification consiste à modéliser le mouvement de l'utilisateur. Ceci équivaut à retrouver l'évolution dans le temps de la *pose* de l'utilisateur, c'est-à-dire la position de chacun de ses membres. De très nombreux travaux ont été proposés dans la littérature concernant la modélisation du mouvement humain. Ceux-ci peuvent être regroupés en trois grandes familles d'approches :

- les approches par apprentissage qui déterminent directement la pose à partir des observations à l'aide d'une fonction apprise à partir d'une base de données d'exemples ;
- les approches par ajustement de modèle qui reposent sur la connaissance d'un modèle articulé représentant l'utilisateur et sur le calcul de la meilleure pose par une maximisation de la vraisemblance entre ce modèle et les données observées ;

– les approches sans information *a priori* qui ne nécessitent aucune connaissance sur le modèle et découvrent automatiquement sa structure articulée.

Nous présentons dans la suite ces différentes familles. Les travaux dans ce domaine étant très nombreux, cet état de l'art ne se veut pas exhaustif et ne donne que les grandes catégories d'approches existantes. Pour un état de l'art plus complet nous renvoyons le lecteur intéressé aux articles très complets de Moeslund *et al.* [Moeslund 01, Moeslund 06].

8.1 Approches par apprentissage

Les approches par apprentissage reposent sur la recherche d'une fonction permettant d'obtenir directement la pose de la personne en fonction de descripteurs sur la forme ou sur les images caractérisant la scène. Cette fonction est obtenue au préalable par apprentissage sur une base de données d'exemples contenant des poses et les descripteurs leur correspondant. Les travaux scientifiques sur ces approches concernent généralement les descripteurs et la méthode d'apprentissage. Beaucoup d'entre-eux utilisent en tant qu'observations des caractérisations de la silhouette. En ce qui concerne la méthode d'apprentissage, une grande variété d'approches existe comme l'utilisation d'analyse en composantes principales (PCA) [Grauman 03] ou de *Relevance Vector Machine* (RVM) [Agarwal 04] qui permettent de trouver un jeu de coefficients pour la fonction recherchée minimisant les erreurs sur les exemples de la base d'apprentissage. La plupart de ces travaux cherchent à résoudre le problème avec une seule caméra et sont ainsi confrontés à des ambiguïtés de position (plusieurs poses peuvent avoir les mêmes descripteurs). Pour résoudre ce problème, certains travaux proposent d'utiliser la cohérence temporelle mais ces techniques deviennent alors inutilisables dans notre contexte car elles nécessitent de travailler sur la séquence complète et pas simplement sur les trames précédant la trame courante. Une autre approche consiste à étendre le problème à plusieurs caméras comme le propose [Grauman 03].

Ces approches fournissent généralement d'assez bons résultats mais dépendent de la qualité de la base d'apprentissage. Elles sont en effet limitées aux mouvements proches de ceux présents dans cette base. Ces exemples sont généralement obtenus en utilisant un générateur de poses, baptisé POSER, permettant de générer l'image d'un modèle à partir d'une pose. Malgré l'automatisation de la génération des exemples, la confection d'une base d'exemples prenant en compte toutes les positions que peut prendre un être humain est un véritable défi (personne marchant sur les mains, saut périlleux, etc...). De plus une telle base de données serait difficilement manipulable tellement le nombre d'exemples serait grand.

8.2 Approches par ajustement de modèle

Une autre famille d'approches repose sur l'ajustement d'un modèle articulé *a priori* avec les données observées. Une mesure de vraisemblance est généralement définie entre le modèle paramétré par la pose et les observations. L'ajustement correspond alors à la maximisation de cette vraisemblance.

Divers modèles articulés ont été proposés représentant généralement la surface ou le volume occupé par l'utilisateur dans l'espace. Certains travaux comme [Carranza 03, Bottino 01, Kakadiaris 00] utilisent un modèle de grande précision du maillage surfacique de l'utilisateur. Ces modèles sont toutefois très difficiles à obtenir car ils doivent correspondre fidèlement à l'utilisateur. D'autres travaux proposent d'utiliser des modèles plus grossiers à base de primitives géométriques simples. Ainsi [Knossow 06, Senior 03, Deutscher 00, Cohen 01] utilisent des cylindres généralisés pour représenter chaque membre. [Cheung 00, Mikić 03] proposent d'utiliser à la place des ellipsoïdes. D'autres primitives géométriques ont été proposées par exemple par [Delamarre 99, Drummond 01, Gavrila 96] cherchant à améliorer la qualité de la représentation de l'utilisateur. Tous ces modèles représentent la forme de l'utilisateur et nécessitent donc de connaître les paramètres caractérisant la forme de chaque membre. Certains travaux comme [Cheung 03] permettent de calculer automatiquement ces paramètres de forme par une phase d'initialisation où l'utilisateur doit réaliser des mouvements spécifiques. Ces techniques restent toutefois assez délicates et nuisent fortement à la flexibilité du système, en particulier pour des applications où n'importe quel utilisateur peut interagir à tout moment (espace public par exemple).

Des solutions plus adéquates cherchent à minimiser l'information de forme dans le modèle *a priori*. Des modèles filaires dits squelettiques peuvent en particulier être utilisés à cette fin. Ce type de modèle ne comporte aucune information volumétrique ou surfacique sur la personne. Ils dépendent ainsi de moins de paramètres de taille spécifiques à chaque utilisateur. De plus les longueurs des membres humains tendent à suivre des lois naturelles – leur rapport entre eux est quasiment universel – alors que les dimensions d'épaisseur peuvent fortement varier au sein d'une population. En comparaison aux approches utilisant un modèle de la forme, peu de travaux ont été proposés sur l'utilisation de modèles squelettiques. Theobalt *et al.* [Theobalt 04] proposent d'ajuster un modèle filaire à l'aide d'informations de suivi des extrémités (pieds, mains et visage) et d'une modélisation volumique de l'enveloppe visuelle. Cependant le modèle n'est pas complètement squelettique car il nécessite des informations d'épaisseur pour les bras et les jambes. Luck *et al.* [Luck 01] proposent une méthode où les squelettes des bras sont ajustés à une modélisation volumique de la partie supérieure du corps de l'utilisateur. Cependant cet approche requiert la connaissance *a priori* de l'épaisseur du torso.

Ces modèles articulés *a priori* de l'utilisateur sont ajustés par rapport à des données

d'observations. Une grande variété de données ont été proposées.

[Gavrila 96, Kakadiaris 00, Carranza 03] utilisent des informations $2D$ comme les silhouettes ou les contours. Le modèle du corps est projeté dans les différents points de vue. L'ajustement est alors réalisé par comparaison de ces projections et des caractéristiques extraites des images acquises. Ces approches posent certaines difficultés. La première concerne le calcul de la projection dans les différents points de vue et la détection des informations images. Si pour le calcul des silhouettes l'utilisation des cartes graphiques est très aisée [Theobalt 03], la projection et l'identification des contours du modèle est en revanche plus délicate. Knossow *et al.* [Knossow 06] proposent une solution analytique au calcul des contours projetés dans le cas de cônes tronqués. Une autre difficulté provient de la nécessité de prendre en compte la visibilité. En effet une information de contour $2D$ d'une image ne doit affecter que la partie visibile du modèle lui correspondant : celle-ci doit donc être identifiée. Enfin l'utilisation d'informations images pose le problème de la fusion des mesures provenant des différents points de vue.

D'autres approches utilisent directement des informations $3D$ sur l'utilisateur. La plupart considèrent le résultat de modélisations des formes par enveloppe visuelle [Bottino 01, Luck 01] ou par stéréo [Delamarre 99]. Ces informations de forme sont particulièrement efficaces pour les approches utilisant des modèles surfaciques comme [Niskanen 05] ou des modèles volumiques [Cheung 00]. En revanche elles s'avèrent moins adaptées en ce qui concerne les modèles squelettiques. Ajuster un modèle filaire à ces données volumiques est difficile de par les disparités de taille entre les différentes parties du corps humain. Ceci explique l'impossibilité pour les approches existantes de se passer complètement de la connaissance de la taille de ces membres.

8.3 Approches sans *a priori*

Plus récemment des travaux ont proposé des méthodes de modélisation du mouvement ne nécessitant pas d'hypothèses sur le modèle *a priori* de l'utilisateur. Elles reposent uniquement sur l'hypothèse que l'objet dans la scène est un objet articulé composé de parties rigides. Ces approches s'attachent alors à identifier ces parties rigides ainsi que les articulations qui composent l'objet.

Cheung *et al.* [Cheung 03] proposent une méthode pour retrouver simultanément la forme et le mouvement de l'utilisateur. Cette méthode repose sur une identification des parties rigides : l'utilisateur "présente" séparément chaque articulation au système pour que celui-ci puisse les identifier. Chu *et al.* [Chu 03] proposent d'identifier le squelette de l'utilisateur par une technique basée sur une isomap. L'identification des articulations nécessite l'accès à l'ensemble de la séquence du mouvement, limitant ainsi son utilisation à un traitement hors-ligne. Plus récemment Brostow *et al.* [Brostow 04] proposent d'extraire le squelette à partir de la forme de l'utilisateur. Cette structure $1D$ permet ensuite d'identifier le mouvement. Ne reposant sur aucun modèle, un grand soin est apporté dans

ce travail sur la qualité de l'extraction de la structure squelettique rendant cette approche très lente.

8.4 Bilan

De très nombreuses méthodes de modélisation du mouvement humain ont été présentées dans la littérature. Trois grandes familles d'approches se dégagent nettement. Les approches par apprentissage calculent à partir d'une base d'exemples une fonction permettant de retrouver directement la pose. Les approches par ajustement de modèles cherchent à maximiser la vraisemblance entre un modèle *a priori* paramétré par la pose et les données observées. Enfin les approches sans *a priori* reposent uniquement sur l'hypothèse que la scène est composée d'objets rigides articulés et cherchent à identifier dans le temps ces parties rigides et les articulations entre-elles.

Les méthodes par apprentissage présentent l'avantage de fournir la pose directement et sont ainsi relativement rapides. Pour ne pas limiter les mouvements de l'utilisateur, il est nécessaire d'utiliser une base de données la plus exhaustive possible. La génération et la gestion d'une telle base reste un problème très difficile : un être humain comportant au moins une vingtaine de degrés de liberté, cette base devrait contenir plusieurs milliards d'exemples pour couvrir l'ensemble des poses possibles. Cette famille d'approches semble donc très efficace lorsque les mouvements sont limités mais apparaît aujourd'hui mal adaptée pour les systèmes d'interaction laissant toute liberté à l'utilisateur.

Les méthodes sans *a priori* fournissent une réponse satisfaisante à ce problème de généricité en ne faisant que très peu d'hypothèses sur la scène (même pas celle de la forme humaine). Ces approches permettent en particulier à un utilisateur d'utiliser un objet rigide dans sa main (cet objet ne peut toutefois pas changer de main). L'utilisation de ces approches dans un système d'interaction pose cependant certaines difficultés. En effet l'identification des articulations nécessitent dans la plupart des cas l'accès à la séquence complète limitant ces approches à des exécutions hors-ligne. De plus il est difficile dans ces méthodes d'identifier précisément à quoi correspond chaque partie rigide (bras, jambe, ...).

Les méthodes par ajustement de modèles semblent aujourd'hui être les mieux adaptées pour répondre au problème de la modélisation du mouvement dans un système d'interaction : elles sont génériques par rapport aux mouvements réalisables, fournissent une identification de chaque membre et peuvent être relativement rapides. Parmi les modèles *a priori* proposés, les modèles squelettiques présentent l'avantage de ne pas dépendre de la forme de l'utilisateur. Ainsi le problème de la modélisation du mouvement se retrouve séparé de celui de la modélisation des formes permettant alors à n'importe

quel utilisateur d'interagir librement. Les méthodes utilisant de tels modèles sont toutefois confrontés au problème d'adéquation entre le modèle et les données d'observation volumiques. En s'inspirant des travaux de Brostow *et al.* , nous proposons dans cette thèse d'utiliser une technique d'extraction d'un squelette filaire $1D$ de la personne à partir du résultat de la modélisation des formes. Ainsi la modélisation du mouvement ne dépend plus de la forme de l'utilisateur.

Principe 9

Nous décrivons dans ce chapitre le principe de notre modélisation du mouvement. Celle-ci repose sur une approche par ajustement d'un modèle squelettique filaire. Cette approche sépare le problème de la modélisation des formes, traité dans la partie II, de celui de la modélisation du mouvement en travaillant sur le squelette $3D$ de l'utilisateur.

La section 9.1 présente le modèle articulé filaire *a priori*. La section 9.2 décrit les données d'observation que nous utilisons lors de l'ajustement du modèle. Le suivi du modèle articulé est détaillé dans la section 9.3.

9.1 Modèle articulé filaire

Dans cette section nous décrivons le modèle articulé *a priori* représentant la pose du corps. A la différence de la grande majorité des approches par ajustement décrites dans la littérature qui utilisent une représentation de la forme de la personne (ellipsoïdes, quadriques, cylindres généralisés, ...), notre modèle consiste en une chaîne cinématique articulaire de segments. Cette structure est parfaitement $1D$ et ne contient aucune information volumique ou surfacique sur l'utilisateur.

Étant donné que les applications interactives ne sont généralement intéressées que par les articulations principales du corps (coudes, épaules, cou, bassin et genoux), nous limitons notre modèle à un ensemble de 12 segments avec ces 9 articulations (voir figure 9.1). Ce modèle comporte ainsi 24 degrés de liberté : 2 par articulations et 6 pour la position et l'orientation du torse. Remarquons que d'autres modèles, plus fidèles à l'anatomie humaine par exemple, peuvent être utilisés pour des applications plus exigeantes comme l'animation graphique. Pour paramétrer les articulations à deux degrés de liberté, nous

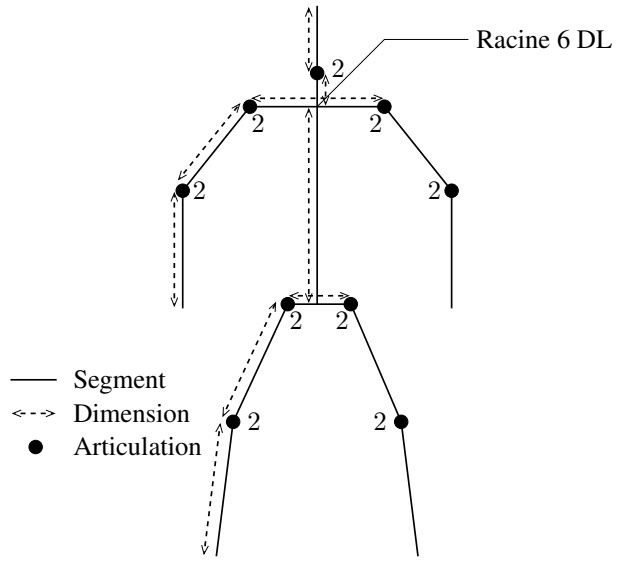


Figure 9.1 *Modèle articulé filaire*

avons choisi une représentation basée sur les angles d'Euler. Pour éviter le problème classique de discontinuités liées à ce type de paramétrage, nous plaçons l'axe de rotation (où les singularités se produisent) dans la direction la moins probable en se basant sur les contraintes angulaires naturelles par exemple. Ceci s'avère suffisant dans la plupart de nos expérimentations. D'autres paramétrages, telles que les quaternions ou celles basées sur une algèbre de Lie, ne donneraient pas forcément de meilleurs résultats étant donné qu'elles représentent des rotations $3D$ complètes (3 degrés de liberté).

9.2 Données d'observation squelettiques

Un autre élément important dans le processus d'ajustement concerne les données d'observation servant à mesurer la pose de l'utilisateur auxquelles le modèle filaire est ajusté. Ces données doivent être choisies en adéquation avec le modèle *a priori* utilisé pour que la mesure de la vraisemblance soit la plus cohérente possible.

L'utilisation de données images $2D$ s'avère très problématique. En plus des difficultés classiques qui leur sont liées telle que la visibilité, une difficulté spécifique à notre modèle filaire s'ajoute. En effet la propriété de squelette n'est pas préservée par projection : l'image projetée du squelette $3D$ ne correspond pas au squelette $2D$ de l'image projetée du modèle. Le squelette de l'utilisateur ne peut donc pas être identifié et mesuré dans les images $2D$.

L'utilisation de données $3D$ s'avère donc nécessaire. Cependant comme nous l'avons décrit, l'utilisation directe de l'information de forme n'est pas adéquat à notre modèle

squelettique. Dans le contexte des approches sans *a priori*, les travaux de Brostow *et al.* [Brostow 04] ont mis en évidence la possibilité d'utiliser une information squelettique extraite de la forme pour l'analyse du mouvement. Cette approche repose sur une *squelettisation*, procédé d'extraction de squelettes filaires $1D$ à partir de la forme $3D$. L'algorithme proposée par Brostow *et al.* permet de récupérer un squelette de grande qualité mais s'avère très lent : il nécessite plusieurs minutes de calcul par trame le rendant inutilisable dans un contexte interactif.

Nous proposons dans cette thèse d'utiliser une squelettisation moins robuste mais nettement plus rapide. Le manque de précision dans l'extraction des informations du squelette est compensé par la suite par l'ajout d'informations *a priori*, c'est-à-dire le modèle filaire humain. La squelettisation a été beaucoup étudiée par la communauté de la géométrie algorithmique et plusieurs définitions ont été proposées. Hastie *et al.* [Hastie 89] proposent par exemple de définir le squelette sous la forme d'une courbe principale (*principal curve*) passant au milieu d'un champ de densité. Il ressort des travaux de cette communauté que la définition de l'axe médian [Blum 67] est la plus populaire et apparaît comme la plus satisfaisante dans notre cas. L'axe médian est défini comme l'ensemble des centres des sphères de taille maximale incluses à l'intérieur du modèle de forme. Une autre façon de voir – équivalente à la précédente – consiste à définir l'axe médian comme l'ensemble des points internes qui possèdent deux points distincts les plus proches sur la surface. Dans le cas d'une surface discrète, le procédé d'obtention d'une approximation discrète de l'axe median est parfois appelé transformée de l'axe médian (*Medial Axis Transform – MAT*). Un inconvénient important de l'axe médian discret vient de sa sensibilité au bruit à la surface du modèle (voir figure 9.2(b)). Des travaux se sont toutefois intéressés à ce problème et ont proposé des techniques prenant en compte le bruit dans les données d'entrée. Attali *et al.* [Attali 96] ont en particulier proposé un tel algorithme sur lequel nous avons basé nos travaux. L'idée est de calculer dans un premier temps l'axe médian discret grâce au diagramme de Voronoi puis d'éliminer les parties correspondant au bruit de la surface que l'on considère comme des points aberrants. L'utilisation du diagramme de Voronoi est motivée par son rapprochement avec la seconde définition de l'axe médian (les centres sont équidistants de deux points de la surface du modèle de la forme de l'utilisateur). L'algorithme procède ainsi :

1. Les centres des cellules du diagrame de Voronoi sont calculés à partir des sommets du maillage d'entrée. Seuls les centres situés à l'intérieur du maillage sont considérés étant donné que nous cherchons l'endo-squelette. (voir figure 9.2(b)).
2. Pour chaque centre C nous récupérons son tétraèdre de Delaunay lui correspondant (P_1, P_2, P_3, P_4) et calculons :

   – son *rayon* $\rho(C) = d(C, P_1)$ et
   – son *angle bissecteur* $\theta(C) = \max_{i \neq j}(\widehat{P_i C P_j})$.

3. En fonction de critères sur un rayon minimal et un seuil sur l'angle bissecteur, les centres correspondant au bruit sont éléminés.

De cette technique résulte un nuage de points $3D$ $\{X_0 \cdots X_n\}$ que nous considérons comme les données squelette d'observation. Le choix d'un seuil pour le rayon ou pour l'angle bissecteur consiste à trouver un compromis entre la qualité du squelette extrait et le nombre de points le définissant. En effet plus les seuils sont importants, meilleure est la qualité mais moins de points sont sélectionnés (voir figure 9.2(d)). En pratique nous choisissons le rayon minimal aux alentours de 4 cm et le seuil sur l'angle bissecteur aux alentours de $160°$ (voir figure 9.2(c)).

Une remarque à ce stade s'impose sur le fait que l'axe médian de données $3D$ n'est pas une courbe filaire, comme c'est le cas en $2D$, mais une surface. En pratique cela n'a pas d'incidence réelle sur notre méthode de suivi du mouvement pour deux raisons principales. Premièrement la largeur de cette surface, dans le cas d'un être humain, est généralement inférieure ou au pire comparable (dans le cas du torse) au bruit des données observées. Deuxièmement le modèle squelettique que l'on souhaite obtenir se situe au milieu de la surface de l'axe médian. Cette structure filaire minimise donc bien les distances par rapport aux données d'observation.

Remarquons que n'importe quelle autre méthode de squelettisation peut être utilisée (amincissement topologique, cartes de distance, ...) mais souvent au détriment de la qualité ou de l'interactivité.

9.3 Suivi du modèle

Nous avons défini dans les sections précédentes notre modèle articulé représentant la pose du corps et les données d'observation extraites. Nous décrivons désormais comment la pose la plus probable de l'utilisateur est retrouvée à partir des données observées à chaque trame. Pour cela nous présentons d'abord un modèle génératif probabiliste qui permet d'expliquer les observations en fonction de la pose du modèle. Nous présentons ensuite comment ce modèle probabiliste est utilisé dans le processus de mise en correspondance entre le modèle et les données observées. Ceci se fait par la recherche du maximum *a posteriori* lié à la formulation probabiliste. Enfin nous décrivons comment le mouvement est suivi dans le temps.

9.3.1 Modèle génératif probabiliste

Pour calculer la pose de l'utilisateur à un instant t, nous devons définir la relation entre la pose du modèle articulé *a priori* et les données d'observation. Une première solution serait de caractériser la similarité entre les points du squelette $\{X_0, \cdots, X_n\}$ et le modèle squelettique S à partir des distances de chaque point au plus proche segment du modèle articulé $s \in S$. La probabilité conjointe serait alors exprimée ainsi :

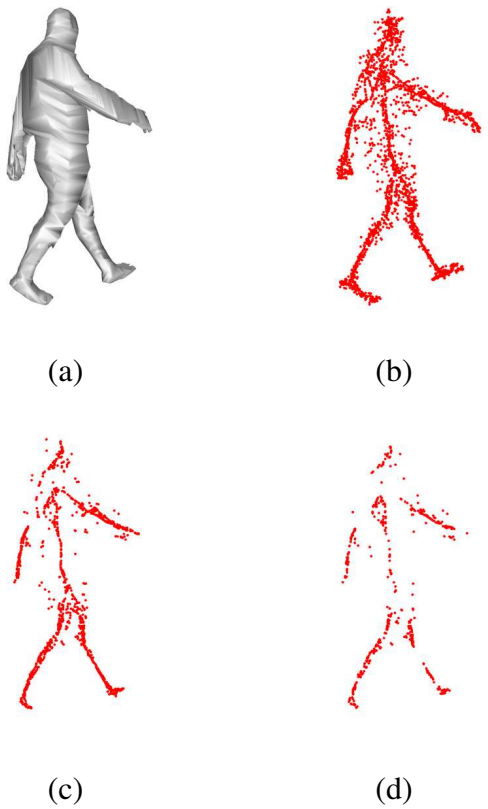


Figure 9.2 *(a) Enveloppe visuelle servant d'entrée à la squelettisation. (b) Les centres des cellules du diagrame de Voronoi produisant un squelette très bruité. (c) Squelettisation après séléction des centres où $r > 4$ cm et $\theta > 160°$: la plupart du bruit est éliminé. (d) Squelettisation après séléction des centres où $r > 5$ cm et $\theta > 170°$: trop de parties sont éliminées.*

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

où $P(X_i|S) = \mathcal{N}(d(X_i, S), \sigma^2)$ et $d(X_i, S) = \min_{s \in S} d(X_i, s)$, avec $d()$ représentant la distance Euclidienne et $\mathcal{N}(x, \sigma^2)$ représentant la loi de distribution gaussienne d'écart type σ.

Cependant maximiser la distribution *a posteriori* correspondante $P(S|\{X_i\})$ entraîne certaines difficultés. L'assignation d'un point à un segment est ammenée à changer au cours du processus d'ajustement générant des incohérences et des discontinuités dans le calcul du gradient. Pour résoudre ce problème, nous introduisons des variables cachées

a_i, une pour chaque point, représentant le segment assigné au point X_i. La probabilité conjointe de la pose du modèle articulé et des observations devient alors :

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

où :

- $P(S)$ correspond à l'information *a priori* sur la pose de l'utilisateur. Dans notre cas nous faisons l'hypothèse d'une distribution uniforme (aucun *a priori*). Cette distribution permettrait par exemple de prendre en compte les contraintes angulaires sur les articulations. Elle permettrait aussi d'inclure des connaissances sur la probabilité des différentes poses : les grand-écarts sont moins probables que la position debout par exemple.
- $P(a_i = j|S)$ représente l'information *a priori* sur l'assignation d'un point à un segment avec la seule connaissance de la pose du modèle. Nous faisons ici l'hypothèse d'une distribution uniforme des points le long des segments. Cet *a priori* est donc proportionnel à la longueur du segment j. Remarquons que dans notre cas, les longueurs des segments sont fixées : cette distribution ne dépend donc pas de la pose.
- $P(X_i|a_i = j\ S)$ représente la probabilité que le point X_i appartienne au membre correspondant au segment j. Nous modélisons cette probabilité par une distribution gaussienne normale $\mathcal{N}(d(X_i, s_j), \sigma_j^2)$. Dans le cas idéal où l'algorithme d'extraction des données squelette est parfait, les σ_j seraient tous équivalents (le bruit ne dépend pas du segment). En pratique cependant, les méthodes de squelettisation induisent plus de bruit au niveau du torse qu'au niveau des bras ou des jambes. Les écart types σ_j sont en pratique fixés aux alentours de 1 cm sauf pour le torse où il avoisine 3 cm.

Trouver la meilleure pose consiste alors à maximiser cet *a posteriori* :

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

Contrairement à la formulation (1), cet *a posteriori* a toutes ses dérivées continues (fonction $\mathcal{C}^\infty$) et peut donc être plus facilement maximiser. De plus cette formulation est plus robuste car elle marginalise sur l'ensemble des assignations points-segments possibles au lieu de ne considérer que l'assignation d'un point avec son segment le plus proche.

9.3.2 Ajustement

Pour calculer le maximum *a posteriori* (MAP) nous utilisons une technique d'Espèrance-Maximisation (*Expectation-Maximization* – EM). Cette technique est classiquement utilisée lorsque la formulation comporte des variables cachées. Nous obtenons ainsi les deux étapes suivantes :

– **L'étape E** consiste à calculer les termes d'espérance $E(a_i = j)$ pour la pose courrement estimée $\overline{S}$:

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

– **L'étape M** consiste à calculer la pose S maximisant :

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

En développant $P(a_i X_i S) = P(S) P(a_i | S) P(X_i | a_i \, S)$, nous remarquons que les deux premiers termes sont constants : $P(S)$ est uniforme par hypothèse et $P(a_i|S)$ ne dépend pas de la pose comme nous l'avons expliqué. Ceci réduit le problème à maximiser :

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

En développant cette formule et en prenant son opposé, le problème est équivalent à minimiser :

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

Cette dernière formule définit un problème de minimisation des moindres carrés. Nous utilisons alors l'algorithme classique de Levenberg-Marquardt, cette méthode de minimisation étant particulièrement adaptée à ce type de problème.

9.3.3 Suivi au cours d'une séquence

Le processus d'ajustement du modèle filaire aux données squelette fournit la pose de l'utilisateur pour une trame donnée. Pour retrouver le mouvement, c'est-à-dire l'enchaînement des poses de l'utilisateur dans le temps, nous devons décrire comment obtenir la pose S_{t+1} pour la trame $t + 1$ à partir des poses estimées des trames précédentes. Ce problème

consiste à prédire une position probable S'_{t+1}. Cette dernière est utilisée comme estimation initiale dans le processus de maximisation de la vraisemblance aboutissant au final à la pose désirée S_{t+1}. Cette prédiction repose généralement sur un modèle dynamique basé sur des hypothèses telles qu'une vitesse ou une accélération contante. Ces modèles sont efficaces lorsque l'on désire suivre les déplacements d'un objet dont la vélocité est stable. Cette condition implique un faible rapport entre les forces appliquées à l'objet et sa masse. Si cette condition est valide pour la position et l'orientation du torse, elle n'est clairement plus vérifiée en ce qui concerne les bras et les jambes : leurs mouvements peuvent être très erratiques. Dans ce cas, suivre le mouvement sans modèle dynamique, c'est-à-dire avec $S'_{t+1} = S_t$, est une bonne solution comme nous le montre nos expériences. Une solution plus satisfaisante serait de considérer que la vélocité des membres est bruité et d'incorporer un modèle de bruit dans le modèle de prédiction. Une manière classique de réaliser cela est d'utiliser par exemple un filtrage particulaire. Nous avons expérimenté cette solution mais sans succès. L'utilisation seule du filtrage particulaire pour effectuer le suivi, c'est-à-dire sans faire intervenir la minimisation précédente, n'aboutit que lorsque le nombre de particules est très grand (plusieurs millions). Ceci se comprend aisément de par le nombre de degrés de liberté du modèle imposant aux particules de représenter une densité de probabilité dans un espace de dimension 24. L'utilisation combinée d'un millier de particules et de la minimisation n'apporte selon nos tests qu'un gain relativement faible (uniquement dans certains cas bien précis) et augmentent significativement le temps de traitement. L'utilisation d'algorithmes de type *Belief-Propagation* pourrait améliorer ces résultats mais au dépend de l'interactivité, ces techniques étant généralement assez lentes.

9.4 Bilan

Ce chapitre a présenté les principes de notre modélisation du mouvement. La principale contribution de cette approche réside en l'ajustement d'un modèle articulé filaire avec des données squelette extraites à partir de la forme de l'utilisateur précédemment calculée.

L'utilisation d'un modèle articulé filaire permet de concentrer l'analyse sur les paramètres intéressants, les degrés de liberté des articulations constituant la pose, sans nécessiter l'estimation simultanée de la forme de la personne. Seuls quelques paramètres de longueur des membres sont nécessaires.

Pour ajuster ce modèle, nous avons proposé d'utiliser des données squelette plus adéquates que des informations de forme ou des données images. Ces données d'observation sont extraites par squelettisation du modèle de la géométrie de l'utilisateur : nous utilisons pour cela une méthode de transformée en l'axe médian prenant en compte le

bruit lié aux données d'entrée. Les données squelette correspondent ainsi à un nuage de point.

Une formulation probabiliste a été proposée pour expliquer les observations en fonction de la pose. Grâce à l'introduction de variables cachées sur l'assignation des points aux segments du modèle, cette formulation est assez robuste aux erreurs d'assignation et permet de retrouver plus facilement le maximum *a posteriori*. Les hypothèses que nous avons formulées sur les formes paramétriques des termes nous ont permis d'effectuer l'ajustement sous la forme d'une minimisation des moindres carrés.

La validation expérimentale de cette approche est présentée dans le chapitre suivant.

Cette approche présente un large éventail d'améliorations possibles. Tout d'abord une idée qui vient naturellement est d'inclure, comme il a été décrit, des connaissances *a priori* sur la probabilité des poses via le terme $P(S)$. La seule prise en compte des contraintes articulaires ne semble pas suffisante au vu de nos expériences à ce sujet : elles ne fournissent une information que dans de très rares cas, très peu de poses étant réellement irréalisables. En revanche l'utilisateur effectue des mouvements naturels en interagissant avec le système et il existe certainement des poses plus fréquentes que d'autres. Cette information serait particulièrement utile pour éviter les minima locaux.

Une autre amélioration pourrait venir de l'intégration de la dynamique au sein de la formulation probabiliste. Ceci permettrait directement d'éviter les écueils liés aux modèles prédictifs comme nous les avons présenté. Cette prise en compte de la dynamique peut être réalisée assez simplement par le changement de $P(S)$ par $P(S_t|S_{t-1})$ décrivant la probabilité d'obtenir la pose S_t à l'instant t connaissant la pose S_{t-1} à l'instant précédent.

Enfin une dernière amélioration envisageable concerne l'intégration de plusieurs données d'observation telles que la couleur du modèle ou encore des informations de plus haut niveau comme la détection de la peau, des pieds/mains ou de la tête.

Tests 10

Pour valider notre modélisation du mouvement nous l'avons implantée et testée sur des séquences synthétiques et plus particulièrement sur des séquences réelles. La section 10.1 présente les séquences utilisées en montrant leurs enjeux. Les résultats du suivi de mouvement sur ces séquences sont présentés dans la section 10.2 et sont discutés dans la section 10.3 sur le plan de la robustesse ainsi que des performances de l'algorithme.

10.1 Séquences de test

Notre approche a été tout d'abord testée sur des séquences de synthèse. Ces données ont été générées par le logiciel Poser qui permet de générer les images selon plusieurs points de vue d'un mouvement d'un être humain. 7 caméras ont été simulées filmant la scène à 30 trames par seconde et avec des images de faible résolution 300×300. A partir des silhouettes générées, les modèles de forme de la personne ont été calculés par la méthode de calcul de l'enveloppe visuelle polyèdrique exacte. Ces séquences ont permis de valider l'approche sur des données parfaites : pas d'erreurs dans la modélisation des formes les silhouettes étant exactes. Le choix d'une faible résolution permet de tester la robustesse de la méthode par rapport à la qualité du maillage reconstruit de l'utilisateur. Nous nous intéressons par la suite à une séquence de 100 trames représentant un homme courant et tournant simultanément.

Les séquences de synthèse ne permettent pas en revanche de valider l'approche en situation réelle. Nous nous sommes donc particulièrement intéressés à la validation sur des séquences réelles. Celles-ci ont été acquises dans la plateforme GrImage en utilisant

le système décrit dans les parties précédentes. La modélisation des formes utilisée correspond au calcul de l'enveloppe visuelle surfacique exacte. La fréquence d'acquisition de ces séquences est de 27 trames par seconde. Deux séquences ont principalement été utilisées pour tester notre approche. La première, baptisée *Marche-En-Rond*, consiste en une personne marchant dans un mouvement circulaire. Cette séquence dure environ 15 secondes et comporte 400 trames, correspondant à deux tours complets autour du cercle. La seconde, baptisée *Coup-De-Pied*, consiste en une action très rapide de coup de pied en l'air réalisé par un professeur de Taekwondo. Elle dure 4 secondes dont seulement 30 trames pour l'action en elle-même. Cette séquence est particulièrement délicate de par la rapidité du mouvement. Ces séquences sont disponibles sur le serveur de séquences de l'équipe PERCEPTION : `https://charibdis.inrialpes.fr`

Les dimensions du modèle *a priori* ont été fixées manuellement avec une erreur d'environ 10%.

10.2 Résultats

Figure 10.1 *Poses calculées pour différentes trames de la séquence synthétique de course générée par le logiciel Poser.*

Les résultats sur la séquence de course synthétique sont présentés figure 10.1. Le suivi du mouvement ne pose pas de difficultés pour cette séquence. Nous pouvons retracer l'évolution des paramètres angulaires des hanches (voir figure 10.2) et remarquer que l'évolution quasi-sinusoïdale donnée par Poser est relativement bien retrouvée. Etant donné que ce logiciel utilise un modèle articulé très différent du nôtre, aucune comparaison numérique n'a pu être réalisée. Nous ne nous sommes donc pas attardés à tester d'avantage notre algorithme sur des séquences de synthèse et avons préferré étudier le comportement de notre méthode sur des séquences réalistes plus ambitieuses.

La figure 10.3 présente les poses de l'utilisateur estimées pour différentes trames de la séquence Marche-En-Rond. La validation de ces résultats a été réalisée manuellement en visualisant chaque trame. Le suivi est dans l'ensemble satisfaisant : seules 6 trames sur les 400 que comportent la séquence sont partiellement erronnées. Ces 6 trames correspondent à deux groupes de 3 trames consécutives, chaque groupe représentant

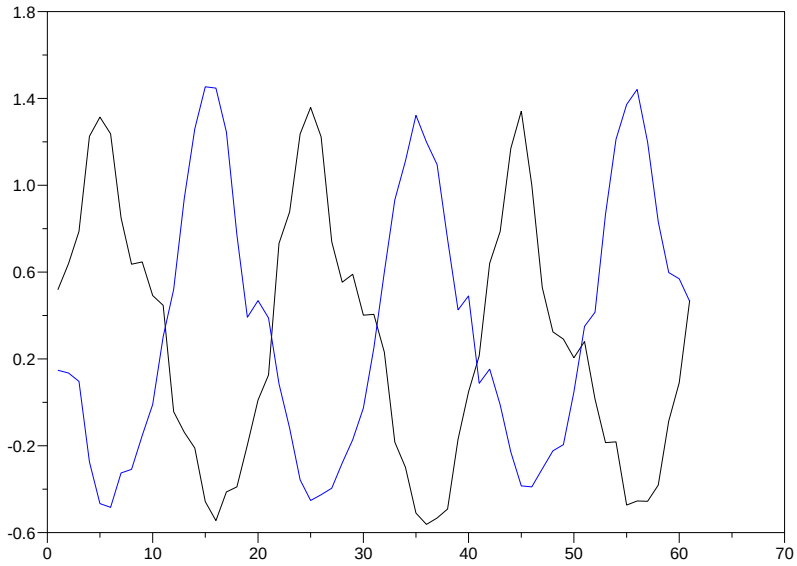


Figure 10.2 *Evolution des paramètres angulaires pour la anche gauche et droite sur la séquence synthétique de course. L'évolution quasi-sinusoïdale fournie par le logiciel Poser est relativement bien retrouvée.*

la même action mais à des instants différents dans la séquence. Dans ces situations, la position du coude estimée se trouve éloignée de sa véritable position : ceci est visible sur la trame 290 de la figure 10.3. Ce phénomème est dû à des problèmes de visibilité qui se traduisent par la présence de points aberrants entre le bras et le torse dans les données squelette. Les segments du bras sont alors assignés en partie à des données du torse, déplaçant le coude vers ce dernier. Remarquons qu'une telle situation pourrait probablement être évitée en ajoutant plus de caméras ou en intégrant la cohérence temporelle, mais là encore au prix d'un surcoût de calcul.

La figure 10.4 présente les poses de l´acteur estimées pour trois différentes trames durant la partie ascendante du coup de pied. Cette séquence a permis de valider le comportement de notre approche face à de grands déplacements entre deux trames consécutives, ou en d'autres termes face aux mouvements rapides comparativement à la fréquence d'acquisition. Comme le montrent ces résultats sans erreurs apparentes de suivi, notre approche se comporte plutôt bien dans ces situations malgré l'absence de modèle dynamique.

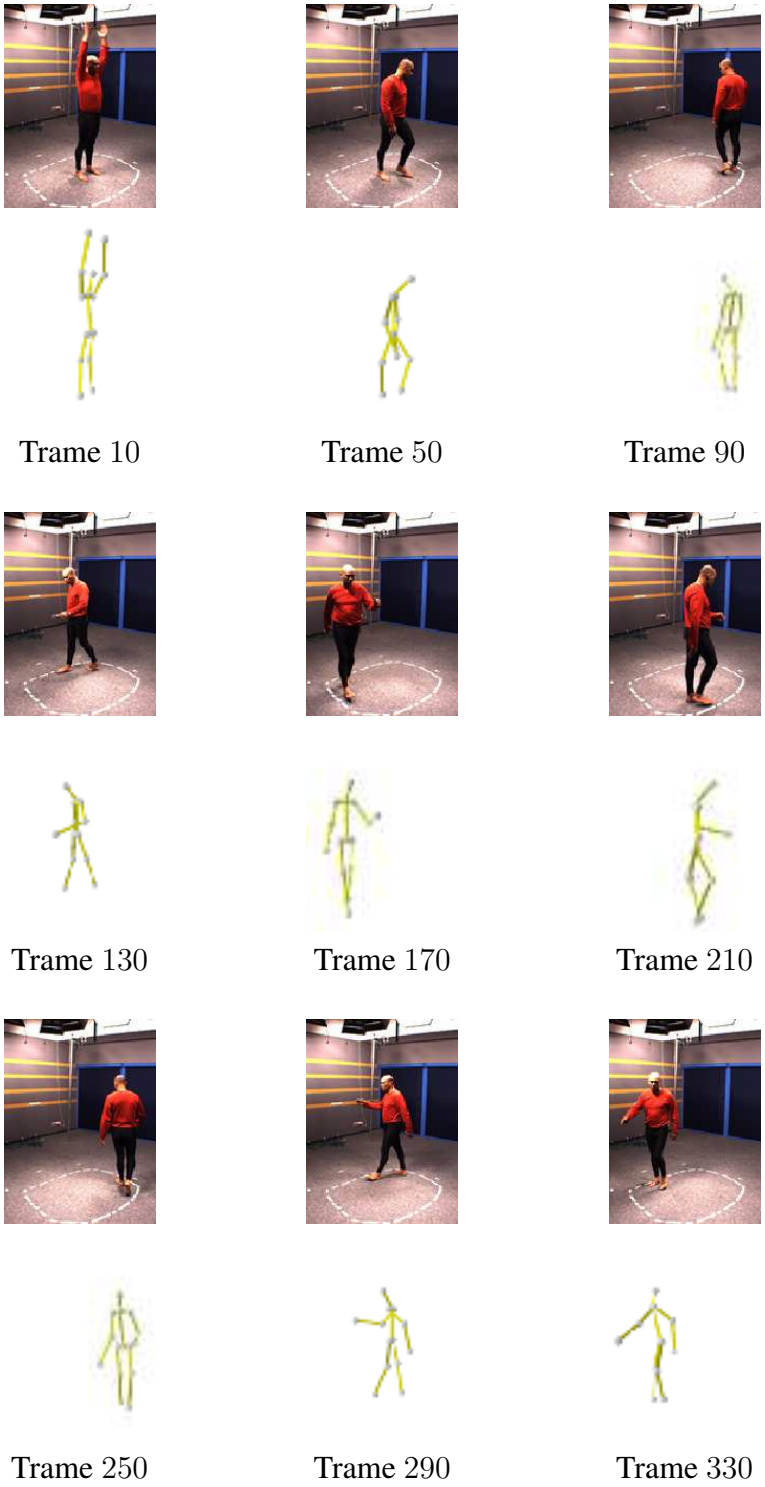


Figure 10.3 *Poses calculées par la modélisation du mouvement pour différentes trames de la séquence de marche.*

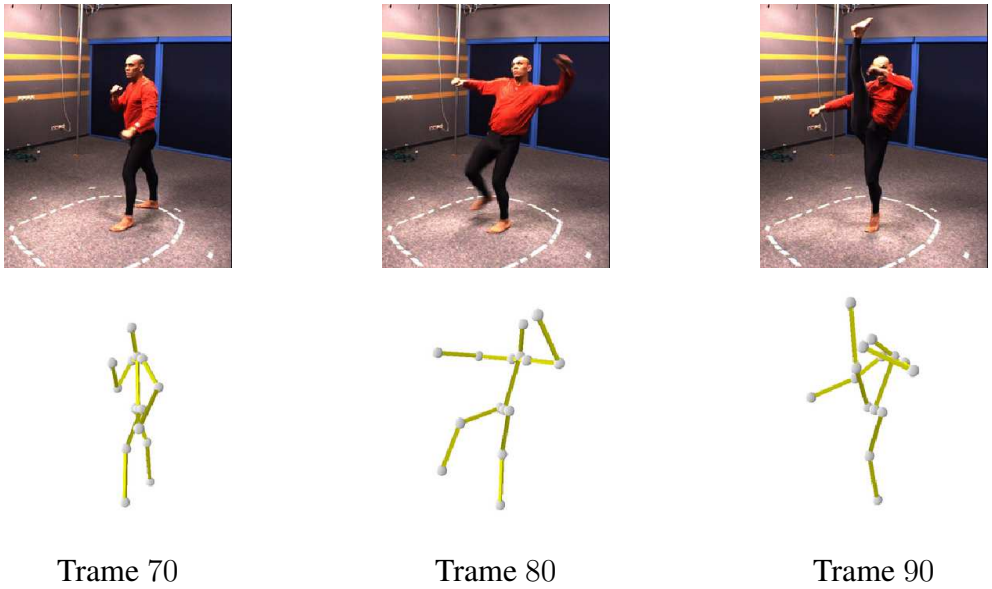


Figure 10.4 *Poses de l'utilisateur calculées pour la séquence du coup de pied très rapide.*

10.3 Discussion

Les expériences sur ces séquences nous ont permis de valider l'approche dans sa globalité. Nous proposons de discuter ces résultats sur le plan de la robustesse et des performances de l'algorithme.

10.3.1 Robustesse

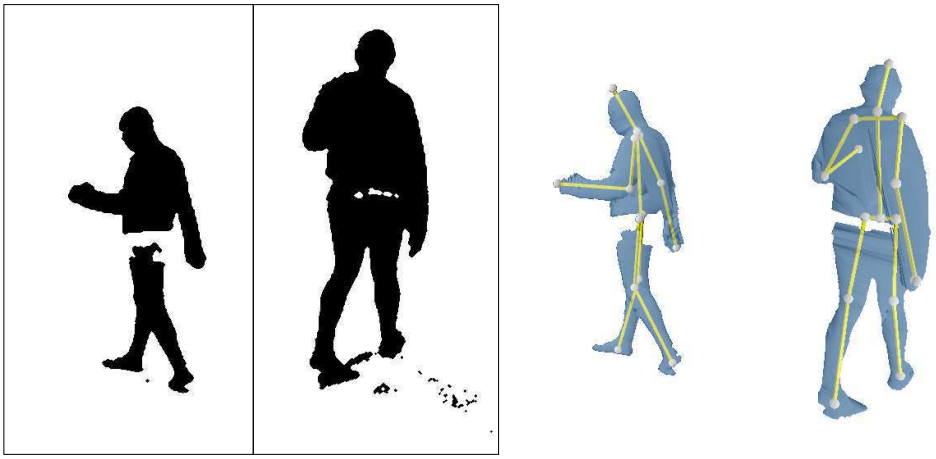


Figure 10.5 *A gauche, exemples de silhouettes bruitées dans la séquence. A droite, le résultat de l'estimation de la pose avec ces silhouettes (2 points de vue différents pour la même trame).*

Un aspect important des algorithmes de modélisation du mouvement humain concerne leur robustesse aux différents types de bruit. Dans notre cas, les principales sources d'erreurs proviennent du bruit dans les données d'entrée ainsi que dans les paramètres du modèle *a priori*. Nous discutons ces deux aspects dans cette section.

Données d'entrée bruitées

Les séquences n'étant pas acquises dans un environnement spécifique tel qu'un studio bleu, les silhouettes obtenues par soustraction de fond peuvent comporter un bruit relativement important. La figure 10.5 présente les silhouettes pour une trame de la séquence Marche-En-Rond. Notre approche est robuste à ces erreurs de plusieurs manières. Premièrement, la modélisation de l'enveloppe visuelle utilisée étant exacte par rapport aux silhouettes d'entrée, aucun bruit n'est rajouté. Cette modélisation présente même l'avantage de filtrer les bruits incohérents dans les différents points de vue : si une camera voit un objet n'existant pas celui-ci sera automatiquement éliminé lors du calcul de l'enveloppe visuelle. Deuxièmement, la méthode de squelettisation proposée prend

en compte le bruit sur la surface reconstruite et permet d'en éliminer une grande partie. Les résultats du suivi présentés à la figure 10.5 valident en pratique la robustesse de notre approche par rapport aux bruits dans les données d'entrée.

Erreurs dans le modèle *a priori*

Pour mesurer la robustesse aux erreurs dans les paramètres du modèle *a priori*, nous avons volontairement introduit du bruit (gaussien) dans les dimensions du modèle utilisé pour la séquence Marche-En-Rond. La modélisation du mouvement se comporte correctement jusqu'à 20% d'erreur : seules les poses de quelques trames se retrouvent partiellement erronnées (moins de 5% des trames sur 10 modèles testés). Pour un bruit plus élevé, le nombre de trames partiellement erronnées augmente rapidement : plus d'une trentaine de trames sur 400 avec 30% de bruit. De plus des erreurs de suivi plus importantes peuvent alors faire échouer le suivi : sur 10 essais de modèles bruités à 30% réalisés deux ont échoué au bout de quelques trames. Notre approche est donc relativement robuste aux erreurs de dimensions du modèle *a priori* mais comporte tout de même certaines limites.

10.3.2 Performance

Un autre aspect très important pour l'utilisation de cette approche de modélisation du mouvement dans un système interactif concerne ses performances. Nous discutons ici des temps de calcul pour les deux étapes principales de notre approche.

Extraction des données squelette

Comme nous l'avons montré dans les premières parties de cette thèse, l'obtention du maillage de la surface de l'utilisateur est réalisable en temps réel avec une latence relativement faible. Tout comme la modélisation des formes hybride, le temps de squelettisation repose essentiellement sur le calcul du diagramme de Voronoi – équivalent au calcul de la tétraédrisation de Delaunay –. Ceci prend environ 80 ms pour traiter 2000 points sur un processeur Opteron 2 GHz. Comme pour la modélisation hybride une parallélisation par flux permet d'obtenir un débit de 30 trames par seconde mais ne réduit pas la latence qui avoisine au total 150 ms. Contrairement à la modélisation hybride, une parallélisation obtenant un résultat approximé peut être réalisée en partitionnant l'espace en zones et en effectuant la squelettisation dans chaque zone indépendamment.

Suivi du mouvement

Dans son implantation actuelle, le suivi du mouvement nécessite environ une seconde par trame. La majorité du temps est utilisée pour calculer les distances entre les points et les segments du modèle *a priori*. Ceci pourrait être réduit en remarquant que seuls les 2 ou 3 segments les plus proches contribuent au calcul en un point donné. Ceci réduirait les temps de calcul par un facteur 5 environ. Remarquons aussi que cette implantation est seulement un prototype expérimental. L'optimisation du code permettrait de réduire

de façon significative les temps de calcul. Enfin la fonction *a posteriori* peut largement bénéficier d'une parallélisation sur plusieurs processeurs, chaque point des données squelette étant indépendantes.

10.4 Bilan

Nous avons testé notre approche sur des séquences réelles acquises dans la plateforme GrImage. Les résultats de suivi du mouvement ne comportent que très peu d'erreurs partielles, démontrant ainsi la validité de notre approche basée sur le squelette.

Ces tests ont aussi pu démontrer la robustesse de notre approche par rapport aux bruits des données d'entrée, principalement liés aux erreurs dans les silhouettes. Cet algorightme présente une relativement grande tolérance vis-à-vis de la précision des estimations des longueurs des membres de l'utilisateur. Ces données peuvent contenir jusqu'à 10% d'erreurs sans trop gêner le suivi du mouvement. Cette tolérance est particulièrement intéressante car elle permet d'envisager l'utilisation d'un modèle unique pour tous les utilisateurs uniquement mis à l'échelle en fonction de leur taille. En effet il semble ressortir des travaux anthropomorphiques, que la seule connaissance de la hauteur d'un être humain soit suffisante pour déterminer la longueur de ses membres avec une précision de 10%. Ceci nécessiterait cependant une étude plus approfondie. En particulier des tests sur une plus grande variété de sujets (hommes, femmes, enfants) serait nécessaire pour valider l'indépendance de notre approche vis-à-vis de la forme de l'utilisateur.

Du point de vue des temps d'exécution, l'implantation actuelle ne permet pas d'envisager son utilisation dans un système interactif. En revanche cette méthode offre plusieurs possibilités d'amélioration que nous avons présentées et qui permettront sans doute d'obtenir une exécution temps réel dans un futur très proche.

V

Conclusion et perspectives

Conclusion et perspectives 11

11.1 Bilan global sur nos travaux

Dans cette thèse nous nous sommes intéressés à l'extraction d'informations $3D$ sur une scène à partir d'images dans le contexte des applications interactives. Les contributions présentées dans ce document touchent plusieurs aspects des systèmes multi-caméras.

Nous avons présenté un système d'interaction permettant de modéliser la forme et le mouvement d'un utilisateur à partir d'images couleur issues de caméras filmant la scène depuis plusieurs points de vue. Ce système se caractérise par une grande flexibilité vis à vis des différents besoins des applications finales : différentes modélisations des formes ont été développées chacune ayant ses spécificités. Les modèles surfaciques sont utiles pour le rendu ou les collisions avec d'autres maillages. Les modèles volumiques sont quant à eux intéressants lorsqu'une représentation de l'occupation de l'espace est nécessaire comme pour les interactions avec une simulation de fluide. Un contrôle entre performance et qualité des données est offert pour chacun de ces algorithmes : ceci permet de s'adapter aux exigences de chaque application.

Grâce à ce système, différentes applications interactives ont été présentées. Ces démonstrations valident en pratique cette approche. Le succès qu'elles ont rencontré auprès du public démontre bien la facilité d'utilisation et l'intérêt d'un tel système pour les interfaces Homme-Machine.

Pour obtenir les meilleurs performances possibles, ce système repose sur une distribution des algorithmes de modélisation des formes. Grâce à cette distribution des

calculs, des modèles surfaciques ou volumiques de bonne précision peuvent être calculés en temps réel avec un débit supérieur à 30 trames par seconde et une latence inférieure à 70 ms. Grâce à l'utilisation de plusieurs processeurs, la précision des modèles est de l'ordre du centimètre voire même inférieure lorsque l'environnement est plus contrôlé (studio bleu par exemple).

Enfin pour ajouter une information de suivi et d'identification aux modèles de formes, nous avons présenté une méthode de modélisation du mouvement humain. Cet algorithme repose sur l'ajustement d'un modèle articulé filaire avec des données squelette extraites de la géométrie de l'utilisateur. Cette approche permet de ne pas inclure de dépendance vis à vis de la forme de la personne en ne prenant pas en compte l'épaisseur des membres du corps. Cette méthode s'avére particulièrement intéressante pour les applications interactives ne nécessitant que très peu d'informations sur l'utilisateur (potentiellement que sa hauteur).

11.2 Parallélisme et interactions : un bon tandem

L'obtention d'informations de qualité tout en maintenant des performances temps réel avec une faible latence n'est possible que grâce à l'utilisation de l'ensemble des ressources de calcul disponibles. Nos travaux de parallélisation de quelques algorithmes de vision par ordinateur ont mis en évidence que ce type de calculs pouvait facilement être distribué et ceci de façon efficace. Ces résultats très encourageants laissent envisager que beaucoup d'autres algorithmes de ce domaine pourraient grandement bénéficier de la parallélisation pour accélérer leur calcul. En collaborant sur d'autres projets, nous avons pu constater que beaucoup d'algorithmes de vision par ordinateur comportent un grand nombre de tâches indépendantes (traitements par voxels, pixels, ...) ou peuvent travailler localement sur une zone de l'espace comme nous l'avons fait pour la modélisation surfacique. L'effort lié à la parallélisation de ces techniques est ainsi relativement faible pour un gain important.

L'étude de la parallélisation des algorithmes de vision par ordinateur est donc particulièrement intéressante et offre de nombreuses possibilités : elle permettrait certainement l'utilisation dans des applications temps réel d'approches jugées aujourd'hui trop lentes. Par exemple la parallélisation de la soustraction de fond permettrait l'utilisation d'algorithmes plus sophistiqués et donc plus performants.

11.3 Les applications envisageables et leurs besoins

Nous avons montré quelques applications interactives réalisables grâce aux données extraites sur la scène. Beaucoup d'autres applications sont envisageables.

L'utilisation du simple modèle visuel, c'est-à-dire la possibilité de réafficher la scène depuis un nouveau point de vue, offre déjà de nombreuses possibilités en particulier dans le domaine des communications ou des media. Cela permet par exemple d'étendre le principe de la visioconférence à la $3D$: l'image habituellement filmée depuis un point de vue statique laisserait place à une image $3D$ calculée en fonction du point d'observation de l'interlocuteur. Ceci permettrait d'améliorer la sensation de présence de la personne éloignée. Un autre domaine pouvant bénéficier d'une telle information concerne le cinéma ou la télévision pour diffuser des films $3D$. Ces applications posent toutefois le problème de la transmission des données $3D$ sur la scène, informations nécessitant beaucoup plus d'espace de stockage ou de bande passante qu'une simple image. Cette problématique fait l'objet de travaux de recherche dans le cadre du projet DALIA.

L'utitilisation de l'information de forme, c'est-à-dire la géométrie $3D$ de la scène permet d'envisager des applications interactives principalement basées sur le principe de la détection de collisions. Nous avons en particulier montré l'utilisation de cette information avec différentes simulations physiques. Cette information de forme peut servir à d'autres applications, par exemple pour déclencher des actions lorsque l'utilisateur "touche" des zones de l'espace. L'intérêt principal de notre approche, par rapport à l'utilisation d'autres capteurs, est de laisser l'utilisateur libre de ses mouvements, des objets qui l'accompagnent. De plus il n'est pas nécessaire de porter un équipement spécifique. Cela est particulièrement intéressant pour des applications où il n'est pas possible d'équiper les utilisateurs. Nous pouvons ainsi imaginer un système permettant de déclencher des actions lorsque une personne passe devant une zone précise d'un bâtiment : extinction d'un appareil électrique lorsqu'un enfant s'en approche par exemple.

Une autre application très intéressante liée à la modélisation de la forme concerne la virtualisation d'objets. En effet il est possible de transformer un objet réel en un objet virtuel grâce à sa forme et son apparence liée aux textures. Cet objet virtualisé peut ensuite faire partie intégrante des simulations en tant qu'objet rigide ou même plus ambitieux en tant qu'objet déformable. Ce mécanisme de virtualisation permet de rendre la frontière entre le réel et le virtuel encore plus flou. Il permet aussi la conception très aisée d'un univers virtuel à base d'objets réels : il suffit de présenter au système l'ensemble des objets composant ce monde.

Si cette information de forme permet de répondre aux besoins d'un très grand nombre d'applications, certaines applications nécessitent toutefois des informations complèmentaires.

Dans le cadre des interactions avec des simulations nous nous sommes rendu compte du fait que l'information de vitesse faisait défaut. En effet notre système a été très récemment utilisé pour interagir avec des objets déformables simulés à l'aide du moteur physique SOFA. Pour la stabilité des calculs celui-ci se base sur des intégrations implicites et nécessite donc la vitesse de déplacement du modèle reconstruit. Cette information permet en outre de transférer de l'énergie cinétique aux objets que l'on collisionne.

L'ajout d'informations de suivi ou d'identification est aussi nécessaire pour certaines applications. Nous avons pu voir que l'identification aussi simple que celle de la tête offrait de nombreuses possibilités comme la simulation d'une nouvelle chevelure. Le suivi de la surface peut quand à lui permettre de modifier l'apparence de la personne : lorsque l'utilisateur plonge une partie de son corps dans un fluide, celle-ci s'en trouve mouillée et garde cet aspect malgré les mouvements de la personne.

Pour permettre des interactions ambitieuses et plus réalistes, l'ajout d'informations en sus de celle de la forme semble nécessaire. Nous avons proposé une première approche pour extraire ces informations mais de nombreux efforts restent à faire pour obtenir une méthode robuste et temps réel.

11.4 Vers les interactions Homme-Machine

Dans cette thèse nous nous sommes focalisés sur le problème de l'extraction d'informations en occultant la manière dont celles-ci sont utilisées. Pour approfondir ces travaux, il est nécessaire de ne plus dissocier l'acquisition de l'utilisation de ces données et d'étudier le mécanisme d'interaction en lui-même. Les paradigmes classiques ne sont plus forcément adaptés et une étude auprès d'un panel d'utilisateurs doit être réalisée pour mesurer l'utilité de ces paradigmes. Les systèmes de vision multi-caméras nécessitent certainement la conception de nouvelles méthodes d'interaction qui leur sont plus adaptées. Il est donc important de se tourner vers le domaine des interactions Homme-Machine pour comprendre comment utiliser au mieux les données extraites pour les interactions.

Un autre point pour lequel il est nécessaire de collaborer avec le domaine des interactions Homme-Machine concerne la qualité des modèles. Dans cette thèse nous avons pris le parti de mesurer celle-ci à l'aide de la précision métrique. En revanche il serait beaucoup plus intéressant de mesurer réellement cette qualité grâce à une étude auprès des utilisateurs. Nous connaissons en particulier pour les robots l'existence d'une zone de précision où l'être humain n'adhère absolument pas à la représentation qui lui est

présentée : ce phénomène est baptisée *Uncanny-Valley*. Il serait intéressant grâce à cette étude de mesurer si notre représentation texturée permet d'aller au-delà de ce phénomène.

Un autre critère de qualité des modèles concerne les performances. La latence mesurée d'environ 100 ms est satisfaisante pour des applications telles que celles que nous avons présentées. Cependant d'autres applications peuvent nécessiter une latence plus faible, quitte à sacrifier la précision des résultats. En ce qui concerne les débits, notre système est limité à l'heure actuelle par les capacités des caméras. Nous avons pu voir avec l'exemple de la simulation de la chevelure qu'une fréquence de 30 trames par seconde peut ne pas être suffisante. L'utilisation de plus de caméras et une synchronisation décalée par jeu de caméras permettrait d'obtenir de plus fortes fréquences d'acquisition et ainsi de palier à ce problème.

Bibliographie

[Agarwal 04] Ankur Agarwal, Bill Triggs. – 3d human pose from silhouettes by relevance vector regression. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Washington, (USA)*, pp. II 882–888, June 2004.

[Allard 04] Jérémie Allard, Valérie Gouranton, Loïck Lecointre, Sébastien Limet, Emmanuel Melin, Bruno Raffin, Sophie Robert. – Flowvr : a middleware for large scale virtual reality applications. – *Proceedings of Euro-par 2004*, Pisa, Italia, August 2004.

[Allard 05] Jérémie Allard, Clément Ménier, Edmond Boyer, Bruno Raffin. – Running large vr applications on a pc cluster : the flowvr experience. – *Immersive Projection Technology*, October 2005.

[Allard 06] Jérémie Allard, Jean-Sébastien Franco, Clément Ménier, Edmond Boyer, Bruno Raffin. – The grimage platform : A mixed reality environment for interactions. – *Proceedings of the 4th International Conference on Computer Vision Systems*, jan 2006.

[Attali 96] D. Attali, A. Montanvert. – Modeling noise for a better simplification of skeletons. – *Proceedings of Third IEEE International Conference on Image Processing, Lausanne (Switzerland)*, 1996.

[Barnett 94] M. Barnett, S. Gupta, D. G. Payne, L. Shuler, R. van de Geijn, J. Watts. – Building a High-Performance Collective Communication Library. – *Supercomputing'94*, Washington D. C., November 1994. IEEE Computer Society Press.

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

[Bertails 05] Florence Bertails, Clément Ménier, Marie-Paule Cani. – A practical self-shadowing algorithm for interactive hair animations. – *Graphics Interface*, May 2005. – Best student paper award.

[Bertails 06] Florence Bertails. – *Simulation de Chevelures Virtuelles*. – PhD. Thesis, Institut National Polytechnique de Grenoble, 2006.

[Blum 67] H. Blum. – A transformation for extracting new descriptors of shape. – W. Walthen-Dunn (édité par), *Models for the Perception of Speech and Visual Form*, pp. 362–380. MIT Press, Cambridge, 1967.

[Borkowski 04] Stanislaw Borkowski, Sherif Sabry, James L. Crowley. – Projector-camera pair : an universal io device for human machine interaction. – *Polish National Robotics Conference KKR VIII*, jun 2004.

[Borovikov 00] Eugene Borovikov, Larry S. Davis. – A distributed system for real-time volume reconstruction. – *CAMP '00 : Proceedings of the Fifth IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'00)*, p. 183, Washington, DC, USA, 2000. IEEE Computer Society.

[Borovikov 03] E. Borovikov, A. Sussman, L. Davis. – A High Performance Multi-Perspective Vision Studio. – *17th Annual ACM International Conference on Supercomputing, San Francisco (USA)*, 2003.

[Bottino 01] Andrea Bottino, Aldo Laurentini. – A silhouette based technique for the reconstruction of human movement. *Computer Vision and Image Understanding*, 83 :79–95, 2001.

[Boyer 97] Edmond Boyer, Marie-Odile Berger. – 3D surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22(3) :219–233, 1997.

[Boyer 03] Edmond Boyer, Jean-Sebastien Franco. – A Hybrid Approach for Computing Visual Hulls of Complex Objects. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, vol. I, pp. 695–701, 2003.

[Boykov 03] Yuri Boykov, Vladimir Kolmogorov. – Computing geodesics and minimal surfaces via graph cuts. – *ICCV '03 : Proceedings of the Ninth IEEE International Conference on Computer Vision*, p. 26, Washington, DC, USA, 2003. IEEE Computer Society.

[Brostow 04] Gabriel J. Brostow, Irfan Essa, Drew Steedly, Vivek Kwatra. – Novel skeletal representation for articulated creatures. – *Proceedings of the 8th European Conference on Computer Vision, Prague, (Czech Republic)*, pp. Vol III : 66–78, 2004.

[Carranza 03] Joel Carranza, Christian Theobalt, Marcus A. Magnor, Hans-Peter Seidel. – Free-viewpoint video of human actors. *Proceedings ACM SIGGRAPH 2003, San Diego (USA)*, 22(2) :569–577, Juillet 2003.

[Cheung 00] German Cheung, Takeo Kanade, Jean-Yves Bouguet, Mark Holler. – A real time system for robust 3d voxel reconstruction of human motions. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, (USA)*, vol. II, pp. 714 – 720, June 2000.

[Cheung 03] G. Cheung, S. Baker, T. Kanade. – Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, 2003.

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

[Chu 03] C.-W. Chu, O. C. Jenkins, M. J. Matarić. – Markerless kinematic model and motion capture from volume sequences. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Madison, (USA)*, pp. 475–482, 2003.

[Cignoni 95] Paolo Cignoni, Domenico Laforenza, Raffaele Perego, Roberto Scopigno, Claudio Montani. – Evaluation of parallelization strategies for an incremental Delaunay triangulator in E**3. *Concurrency, Pract. Exp.*, 7 :61–80, 1995.

[Cipolla 92] Roberto Cipolla, Andrew Blake. – Surface Shape from the Deformation of Apparent Contours. *International Journal of Computer Vision*, 9 :83–112, 1992.

[Cohen 01] Isaac Cohen, GÃ©rard Medioni, Haisong Gu. – Inference of 3d human body posture from multiple cameras for vision-based user interface. – *5th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando*, 2001.

[Corba95] Object management group - the common object request broker : Architecture and specification, v. 2.0, July 1995. Document formal/97-02-25.

[Cotting 04] Daniel Cotting, Martin Naef, Markus Gross, Henry Fuchs. – Embedding imperceptible patterns into projected images for simultaneous acquisition and display. – *Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pp. 100–109, 2004.

[Crispell 06] Daniel Crispell, Douglas Lanman, Peter G. Sibley, Yong Zhao, Gabriel Taubin. – Beyond silhouettes : Surface reconstruction

using multi-flash photography. – *Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2006)*, June 2006.

[Crowley 95] J. Crowley, F. Berard, J. Coutaz. – Finger tracking as an input device for augmented reality. – *Proceedings of the International Workshop on Automatic Face and Gesture Recognition (IWAFGR '95)*, June 1995.

[Darrel 95] Trevor Darrel, Alex P. Pentland. – Attention-driven expression and gesture analysis in an interactive environment. – *Proceedings of the International Workshop on Automatic Face and Gesture Recognition (IWAFGR '95)*, pp. 135–140, June 1995.

[Debevec 96] Paul E. Debevec, Camillo J. Taylor, Jitendra Malik. – Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-Based Approach. – *ACM Computer Graphics (Proceedings SIGGRAPH)*, pp. 11–20, 1996.

[DebledRennesson 95] Isabelle Debled-Rennesson, Jean-Pierre Reveillès. – A linear algorithm for segmentation of digital curves. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(4) :635–662, 1995.

[DebledRennesson 04] Isabelle Debled-Rennesson, Salvatore Tabbone, Laurent Wendling. – Fast Polygonal Approximation of Digital Curves. – *Proceedings of the 17th International Conference on Pattern Recognitio)*, vol. I, pp. 465–468, 2004.

[Delamarre 99] Quentin Delamarre, Olivier D. Faugeras. – 3d articulated models and multi-view tracking with silhouettes. – *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, (Greece)*, pp. 716–721, 1999.

[delaRivière 05] Jean-Baptiste de la Rivière. – *Suivi Vidéo de Mouvements pour l'Interaction*. – PhD. Thesis, Université Bordeaux 1, June 2005.

[Denis 03] A. Denis, C. Pérez, T. Priol. – Achieving portable and efficient parallel CORBA objects. *Concurrency and Computation : Practice and Experience*, 15(10) :891–909, August 2003.

[Deutscher 00] Jonathan Deutscher, Andrew Blake, Ian Reid. – Articulated body motion capture by annealed particle filtering. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, (USA)*, pp. 2126–2133, 2000.

[Drummond 01] Tom Drummond, Roberto Cipolla. – Real-time tracking of highly articulated structures in the presence of noisy measurements. –

Proceedings of the 8th International Conference on Computer Vision, Vancouver, (Canada), pp. 315–320, 2001.

[Dyer 01] C. R. Dyer. – Volumetric scene reconstruction from multiple views. *Foundations of Image Understanding*, éd. par L. S. Davis, pp. 469–489. – Kluwer, Boston, 2001.

[Fob] Ascension technology - flock of birds. http://www.ascension-tech.com/products/flockofbirds.php.

[Franco 03] Jean-Sebastien Franco, Edmond Boyer. – Exact Polyhedral Visual Hulls. – *Proceedings of the British Machine Vision Conference, Norwich (UK)*, pp. 329–338, Septembre 2003.

[Franco 04] Jean-Sébastien Franco, Clément Ménier, Edmond Boyer, Bruno Raffin. – A distributed approach for real-time 3d modeling. – *CVPR Workshop on Real-Time 3D Sensors and their Applications*, june 2004.

[Franco 05] Jean-Sebastien Franco. – *Modélisation tridimensionnelle à partir de silhouettes*. – PhD. Thesis, INPG, December 2005.

[Fua 91] Pascal Fua. – *A parallel stereo algorithm that produces dense depth maps and preserves image features*. – Rapport de recherche, INRIA, Janvier 1991.

[Gavrila 96] Dariu M. Gavrila, Larry S. Davis. – 3-d model-based tracking of humans in action : a multi-view approach. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, San Francisco)*, pp. 73–80. IEEE Computer Society, 1996.

[Gokturk 04] S. Burak Gokturk, Hakan Yalcin, Cyrus Bamji. – A time-of-flight depth sensor - system description, issues and solutions. – *CVPRW '04 : Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3*, p. 35, Washington, DC, USA, 2004. IEEE Computer Society.

[Gong 05] Minglun Gong, Yee-Hong Yang. – Near real-time reliable stereo matching using programmable graphics hardware. – *CVPR '05 : Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pp. 924–931, Washington, DC, USA, 2005. IEEE Computer Society.

[Graham 66] R. L. Graham. – Bound for certain multiprocessing anomalies. *Bell System Tech. J.*, pp. 1563–1581, 1966.

[Grauman 03] Kristen Grauman, Gregory Shakhnarovich, Trevor Darrell. – Inferring 3d structure with a statistical image-based shape model.

– *Proceedings of the 9th International Conference on Computer Vision, Nice, (France)*, pp. 641–648, 2003.

[Graux 04] Oliver Graux, Tim Pullen, Graham A. Thomas. – A combined studio production system for 3-d capturing of live action and immersive actor feedback. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(3) :370–380, Mars 2004.

[Gross 03] Markus Gross, Stephan Wuermlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlk, Andrew Vande Moere, Oliver Staadt. – Blue-c : A spatially immersive display and 3d video portal for telepresence. – *Proceedings of ACM SIGGRAPH 2003, San Diego (USA)*, San Diego, 2003.

[Hasenfratz 03] Jean-Marc Hasenfratz, Marc Lapierre, Jean-Dominique Gascuel, Edmond Boyer. – Real-time capture, reconstruction and insertion into virtual world of human actors. – *Vision, Video and Graphics*, pp. 49–56. Eurographics, Elsevier, 2003.

[Hasenfratz 04] Jean-Marc Hasenfratz, Marc Lapierre, François Sillion. – A real-time system for full body interaction with virtual worlds. *Eurographics Symposium on Virtual Environments*, pp. 147–156, 2004.

[Hastie 89] Trevor Hastie, Werne Stuetzle. – Principal curves. *Journal of the American Statistical Association*, 84 :502–516, 1989.

[Hilton 04] Adrian Hilton, Jonathan Starck. – Multiple view reconstruction of people. – *3DPVT '04 : Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pp. 357–364, Washington, DC, USA, 2004. IEEE Computer Society.

[Horprasert 99] Thanarat Horprasert, David Harwood, Larry S. Davis. – A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection . – *IEEE ICCV'99 frame-rate workshop*, 1999.

[Intersense] Intersense. – Inertiacube. http://www.isense.com.

[Ishii 94] Masahiro Ishii, Masanori Nakata, Makoto Sato. – Networked spidar : A networked virtual environment with visual, auditory, and haptic interactions. *Presence : Teleoperators and Virtual Environments*, 3(4) :351–359, 1994.

[Kakadiaris 00] Ioannis Kakadiaris, Dimitris Metaxas. – Model-based estimation of 3d human motion. *IEEE Transactions on PAMI*, 22(12) :1453–1459, december 2000.

[Kanade 97] Takeo Kanade, Peter Rander, P.J. Narayanan. – Virtualized reality : Constructing virtual worlds from real scenes. *IEEE Multimedia, Immersive Telepresence*, 4(1) :34–47, January 1997.

[Karaman 05] Mustafa Karaman, Lutz Goldmann, Da Yu, Thomas Sikora. – Comparison of static background segmentation methods. – *Visual Communications and Image Processing (VCIP '05)*, Beijing, China, juillet 2005.

[Kelshikar 03] Nikhil Kelshikar, Xenophon Zabulis, Jane Mulligan, Kostas Daniilidis, Vivek Sawant, Sudipta Sinha, Travis Sparks, Scott Larsen, Herman Towles, Ketan Mayer-Patel, Henry Fuchs, John Urbanic, Kathy Benninger, Raghurama Reddy, Gwendolyn Huntoon. – Real-time terascale implementation of tele-immersion. – *Proceedings of the Terascale Performance Analysis Workshop int conjonction with ICCS 2003*, 2003.

[Knossow 06] David Knossow, Remi Ronfard, Radu P. Horaud, Frédéric Devernay. – Tracking with the kinematics of extremal contours. – P.J. Narayanan, Shree K. Nayar, Heung-Yeung Shum (édité par), *Computer Vision – ACCV 2006*, LNCS, pp. 664–673, Hyderabad, India, January 2006. Springer.

[Koenderink 84] Jan J. Koenderink. – What Does the Occluding Contour Tell us About Solid Shape ? *Perception*, 13 :321–330, 1984.

[Kohout 03] Josef Kohout, Ivana Kolingerová. – Parallel delaunay triangulation in e3 : make it simple. *Visual Computer*, 19(7-8) :532–548, December 2003.

[Kohout 05] Josef Kohout. – *Delaunay Triangulation in Parallel and Distributed Environment*. – Univerzitní 8, 306 14 Pillsen, Czech Republic, PhD. Thesis, University of West Bohemia, May 2005.

[Krueger 85] Myron W. Krueger, Thomas Gionfriddo, Katrin Hinrichsen. – Videoplace – an artificial reality. – *CHI '85 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 35–40. ACM Press, 1985.

[Kry 06] Paul G. Kry, Dinesh K. Pai. – Interaction capture and synthesis. *Proceedings of ACM SIGGRAPH 2006*, 25(3) :872–880, 2006.

[Labatut 06] Patrick Labatut, Renaud Keriven, Jean-Philippe Pons. – Fast level set multi-view stereo on graphics hardware. – *Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2006)*, June 2006.

[Laidlaw 86] David H. Laidlaw, W. Benjamin Trumbore, John F. Hughes. – Constructive solid geometry for polyhedral objects. – *SIG-*

GRAPH '86 : Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 161–170, New York, NY, USA, 1986. ACM Press.

[Laurentini 94] A. Laurentini. – The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on PAMI*, 16(2) :150–162, février 1994.

[Lazebnik 01] Svetlana Lazebnik, Edmond Boyer, Jean Ponce. – On How to Compute Exact Visual Hulls of Object Bounded by Smooth Surfaces. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Kauai, (USA)*, vol. I, pp. 156–161, December 2001.

[Levoy 00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, Duane Fulk. – The digital michelangelo project : 3D scanning of large statues. – Kurt Akeley (édité par), *Siggraph 2000, Computer Graphics Proceedings*, pp. 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[Li 03] Ming Li, Marcus Magnor, Hans-Peter Seidel. – Hardware-accelerated visual hull reconstruction and rendering. – *Proceedings of Graphics Interface'2003*, Halifax, Canada, 2003.

[Li 04] Ming Li, Marcus Magnor, Hans-Peter Seidel. – A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls. – *GI '04 : Proceedings of the 2004 conference on Graphics interface*, pp. 41–48, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[Luck 01] J. Luck, D.Small, C. Q. Little. – Real-time tracking of articulated human models using a 3d shape-from-silhouette method. – *Roboton Vision*, pp. 19–26, 2001.

[Margery 02] D. Margery, B. Arnaldi, A. Chauffaut, S. Donikian, T. Duval. – Openmask : {Multi-Threaded | Modular} animation and simulation {Kernel | Kit } : a general introduction. – Simon Richir, Paul Richard, Bernard Taravel (édité par), *VRIC 2002 Proceedings*, pp. 101–110. ISTIA Innovation, June 2002.

[Martin 83] W.N. Martin, J.K. Aggarwal. – Volumetric description of objects from multiple views. *IEEE Transactions on PAMI*, 5(2) :150–158, 1983.

[Massie 94] Thomas H. Massie, J. K. Salisbury. – The phantom haptic interface : A device for probing virtual objects. – *ASME International Mechanical Engineering Congress and Exhibition*, pp. 295–300, 1994.

[Matsuyama 04] Takashi Matsuyama, Xiaojun Wu, Takeshi Takai, Shohei Nobuhara. – Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Computer Vision and Image Understanding*, 96(3) :393–434, 2004.

[Matusik 00] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Leonard McMillan, Steven Gortler. – Image Based Visual Hulls. – *Proceedings of ACM SIGGRAPH 2000*, pp. 369–374, 2000.

[Matusik 01] Wojciech Matusik, Chris Buehler, Leonard McMillan. – Polyhedral Visual Hulls for Real-Time Rendering. – *Eurographics Workshop on Rendering*, 2001.

[Ménier 04] Clément Ménier, Jérémie Allard, Jean-Sébastien Franco, Bruno Raffin, Edmond Boyer. – Marker-less real time 3d modeling for virtual reality. – *Immersive Projection Technology*, march 2004.

[Ménier 05] Clément Ménier, Jean-Sébastien Franco, Edmond Boyer, Bruno Raffin. – Modélisation tri-dimensionnelle temps-réel et distribuée. – *Actes des Journées ORASIS*, May 2005.

[Ménier 06] Clément Ménier, Edmond Boyer, Bruno Raffin. – 3d skeleton-based body pose recovery. – *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission, Chapel Hill (USA)*, june 2006.

[Mikić 03] Ivana Mikić, Mohan Trivedi, Edward Hunter, Pamela Cosman. – Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3) :199–223, 2003.

[Milne 96] A. D. Milne, J. A. Johnson D. G. Chess, G. J. W. King. – Accuracy of an electromagnetic tracking device : A study of the optimal operating range and metal interference. *Journal of Biomechanics*, 29(6) :791–793, June 1996.

[Moeslund 01] Thomas B. Moeslund, Erik Granum. – A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3) :231–268, March 2001.

[Moeslund 06] Thomas B. Moeslund, Adrian Hilton, Volker Kruger. – A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 103(2-3) :90–126, November 2006.

[Mpi94] University of Tennessee, Knoxville, Tennessee. – *Message Passing Interface Forum - MPI : A Message–Passing Interface Standard*, 1994.

[Niem 94] W. Niem. – Automatic Modeling of 3D Natural Objects from Multiple Views. – *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production, Hamburg, Germany*, 1994.

[Niskanen 05] Matti Niskanen, Edmond Boyer, Radu P. Horaud. – Articulated motion capture from 3-d points and normals. – Torr Clocksin, Fitzgibbon (édité par), *British Machine Vision Conference*, vol. 1, pp. 439–448, Oxford, UK, September 2005. BMVA, British Machine Vision Association.

[O'Hagan 97] Rochelle O'Hagan, Alexander Zelinsky. – Finger track - a robust and real-time gesture interface. – *Australian Joint Conference on Artificial Intelligence*, pp. 475–484, 1997.

[Qhull] Qhull : Convex hull, delaunay triangulation and voronoi diagram library. http://www.qhull.org.

[Reunanen 95] Markku Reunanen, Karri Palovuori, Tommi Ilmonen, Wille Mäkelä. – Näprä – affordable fingertip tracking with ultrasound. – *Eurographics Symposium on Virtual Environments*, pp. 51–58, October 1995.

[Rusinkiewicz 02] Szymon Rusinkiewicz, Olaf Hall-Holt, Marc Levoy. – Real-time 3d model acquisition. – *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 438–446, New York, NY, USA, 2002. ACM Press.

[Scharstein 02] Daniel Scharstein, Richard Szeliski. – A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3), April-June 2002.

[Seitz 06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski. – A comparison and evaluation of multi-view stereo reconstruction algorithms. – *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, New York, (USA)*, vol. 1, pp. 519–528, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[Senior 03] Andrew Senior. – Real-time articulated human body tracking using silhouette information. – *proceedings IEEE Workshop on Visual Surveillance/PETS, Nice, France*, october 2003.

[Shen 04] J. Shen. – Motion detection in color image sequence and shadow elimination. – *Proceedings of SPIE ; Visual Communications and Image Processing 2004.*, vol. 5308, pp. 731–740, 2004.

[Singhal 00] S. Singhal, M. Zyda. – *Networked Virtual Environments - Design and Implementation*. – ACM Press Books, 2000.

[Slabaugh 01] G. Slabaugh, B. Culbertson, T. Malzbender, R. Schafe. – A Survey of Methods for Volumetric Scene Reconstruction from Photographs. – *International Workshop on Volume Graphics*, 2001.

[Soares 07] Luciano Soares, Clément Ménier, Bruno Raffin, Jean-Louis Roch. – Parallel adaptive octree carving for real-time 3d modeling. – *IEEE Virtual Reality*, 2007.

[Szeliski 93] R. Szeliski. – Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing*, 58(1) :23–32, 1993.

[Theobalt 03] Christian Theobalt, Joel Carranza, Marcus A. Magnor, Hans-Peter Seidel. – A parallel framework for silhouette-based human motion capture. – *Proceedings of Vision, Modeling and Visualization 2003*, pp. 207–214, Munich, Germany, 2003.

[Theobalt 04] C. Theobalt, M. Magnor, P. Schüler, 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, octobre 2004. – Pacific Graphics 2002.

[Valiant 90] L. G. Valiant. – A Bridging Model for Parallel Computation. *Communications of the ACM*, 33(8) :103–111, August 1990.

[Wada 00] Toshikazu Wada, Xiaojun Wu, Shogo Tokai, Takashi Matsuyama. – Homography based parallel volume intersection : Toward real-time volume reconstruction using active cameras. – *CAMP '00 : Proceedings of the Fifth IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'00)*, p. 331, Washington, DC, USA, 2000. IEEE Computer Society.

[Waschbüsch 05] Michael Waschbüsch, Stephan Würmlin, Daniel Cotting, Filip Sadlo, Markus Gross. – Scalable 3d video of dynamic scenes. *The Visual Computer (Special Issues for Pacific Graphics 2005)*, 21(8–10) :629–638, September 2005.

[Wu 06] Xiaojun Wu, Osamu Takizawa, Takashi Matsuyama. – Parallel pipeline volume intersection for real-time 3d shape reconstruction on a pc cluster. – *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, Washington, DC, USA, 2006. IEEE Computer Society.

[Xsens] Xsens. – Motion tracker. http://www.xsens.com.

[Zelinsky 96] Alexander Zelinsky, Jochen Heinzmann. – Real-time visual recognition of facial gestures for human computer interaction. – *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 351–356, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1996. IEEE.

[Zhang 99] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, Mubarak Shah. – Shape from shading : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8) :690–706, 1999.