

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : INFORMATIQUE

Présentée et soutenue publiquement par

Ludovic BLACHE

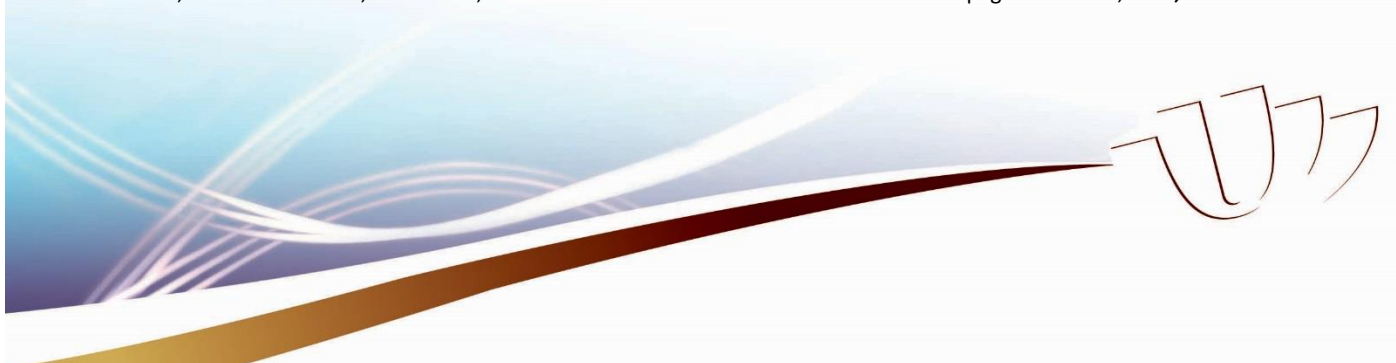
Le 20 avril 2016

REPRÉSENTATION DYNAMIQUE DE MODÈLES D'ACTEURS ISSUS DE RECONSTRUCTIONS MULTI-VUES

Thèse dirigée par **Céline LOSCOS et Laurent LUCAS**

JURY

M. Edmond BOYER,	, Directeur de Recherche,	INRIA,	, Président
M. Pierre ALLIEZ,	, Directeur de Recherche,	INRIA,	, Rapporteur
M. Loïc BARTHE,	, Professeur,	à l'Université de Toulouse 3 Paul Sabatier,	, Rapporteur
M. Mathieu DESBRUN,	, Professeur,	CALTECH, USA,	, Examineur
Mme. Céline LOSCOS,	, Professeur,	à l'Université de Reims Champagne-Ardenne,	, Examineur
M. Laurent LUCAS,	, Professeur,	à l'Université de Reims Champagne-Ardenne,	, Examineur



Abstract

4D multi-view reconstruction technologies are more and more used in media production due to their abilities to produce a virtual clone of an actor from a simple video acquisition performed by a set of multi-viewpoint cameras. This approach is a major advance for the composition of animations which mix virtual and real images, and also offers new possibilities for the rendering of such complex hybrid scenes. The work described in this thesis takes parts in the RECOVER 3D project which aims at developing an innovative industrial framework for TV production, based on multi-view reconstruction, from studio acquisition to broadcasting. The major drawback of the methods used in this context is that they are not adapted to the reconstruction of dynamic scenes. The output are time series which describe the successive poses of the actor, figured as a sequence of static objects. The goal of this thesis is to transform these initial results into a dynamic 3D object where the actor is figured as an animated character. The research detailed in this manuscript presents two main contributions. The first one is centered on the computation of a motion flow which represents the displacements occurring in the reconstructed scene between two consecutive poses. The second one presents a mesh animation process that leads to the animation of a 3D model from one pose to another, following the motion flow. This two-step operation is repeated throughout the entire pose sequence to finally obtain a single animated mesh that matches the evolving shape of the reconstructed actor. Results show that our method is able to produce a temporally consistent mesh animation from various sequences of visual hulls.

Keywords: multi-view reconstruction, animation, mesh deformation, motion flow, as-rigid-as-possible, dynamic scene, volume, voxel matching, temporal consistency.

Résumé

Les technologies de reconstruction multi-vues permettent de réaliser un clone virtuel d'un acteur à partir d'une simple acquisition vidéo réalisée par un ensemble de caméras à partir de multiples points de vue. Cette approche offre de nouvelles opportunités dans le domaine de la composition de scènes hybrides mélangeant les images réelles et virtuelles. Cette thèse a été réalisée dans le cadre du projet RECOVER 3D dont l'objectif est de développer une chaîne de production TV complète, de l'acquisition jusqu'à la diffusion, autour de la reconstruction multi-vues. Cependant la technologie utilisée dans ce contexte est mal adaptée à la reconstruction de scènes dynamiques. En effet, la performance d'un acteur est reproduite sous la forme d'une séquence d'objets 3D statiques qui correspondent aux poses successives du personnage au cours de la capture vidéo. L'objectif de cette thèse est de développer une méthode pour transformer ces séquences de poses en un modèle animé unique. Les travaux de recherches menés dans ce cadre sont répartis en deux étapes principales. La première a pour but de calculer un champ de déplacements qui décrit les mouvements de l'acteur entre deux poses consécutives. La seconde étape consiste à animer un maillage suivant les trajectoires décrites par le champ de mouvements, de manière à le déplacer vers la pose suivante. En répétant ce processus tout au long la séquence, nous parvenons ainsi à reproduire un maillage animé qui adopte les poses successives de l'acteur. Les résultats obtenus montrent que notre méthode peut générer un modèle temporellement cohérent à partir d'une séquence d'enveloppes visuelles.

Mots-clés: reconstruction multi-vues, animation, déformation de maillage, champ de déplacements, as-rigid-as-possible, scène dynamique, volume, appariement de voxels, cohérence temporelle.

Remerciements

Je remercie tout d'abord Pierre Alliez et Loïc Barthe pour avoir accepté de rapporter mon mémoire de thèse. J'adresse également mes remerciements aux autres membres du jury, Mathieu Desbrun et Philippe Souchet, ainsi qu'à Edmond Boyer pour avoir présidé ce jury.

Je tiens ensuite à remercier mes directeurs de thèse, Laurent Lucas et Céline Loscos, pour m'avoir donné l'opportunité de travailler sur ce sujet de thèse ainsi que pour la confiance et la liberté qu'ils m'ont accordées tout au long de ces travaux.

Je souhaite aussi remercier toutes les personnes avec qui j'ai eu l'occasion de travailler durant ces années de thèse. Merci à Jacques Peyrache, Philippe Souchet, Cédric Niquin et l'ensemble du personnel d'XD Productions pour leur aide, leur accueil et leur gentillesse qui ont fait du projet RECOVER3D une expérience captivante. Un grand merci à Mathieu Desbrun et Pierre Alliez pour m'avoir accueilli dans leurs équipes. Vos conseils avisés et les conversations enrichissantes que nous avons partagées ont été essentiels pour mener à bien ces travaux.

Je remercie également les membres du CReSTIC et du département Informatique de l'IUT RCC grâce à qui ces quatre années passées à l'URCA ont été aussi agréables que formatrices. Je souhaite remercier particulièrement Lara, Muhannad, Jennifer, Daniel, Raissel, Francisco, Francky, Exavérine, Jonathan, Ulysse et tous mes camarades thésards, les collègues de la salle machine, et plus largement toutes les personnes que j'ai connues à Reims et qui sont devenus des amis très chers avec qui j'ai partagé des moments inoubliables. La place me manque pour tous les nommer mais je sais qu'ils se reconnaîtront.

Je remercie enfin mes parents grâce à qui j'ai eu la chance de poursuivre mes études en toute sérénité.

Table des matières

1	Introduction	5
1.1	Contexte	9
1.2	Reconstruction multi-vues	10
1.2.1	Aperçu	10
1.2.2	Différentes approches de reconstruction	12
1.2.3	Techniques usuelles	12
1.3	RECOVER 3D	14
1.3.1	Objectifs	14
1.3.2	Processus de reconstruction	16
1.4	Problématique	18
1.5	Objectifs	20
1.6	Données d'entrée	21
1.6.1	Jeux de données	22
1.7	Contributions	22
1.8	Plan	24
2	Etat de l'art	25
2.1	Introduction	29
2.2	Systèmes de reconstruction	29
2.2.1	Systèmes basés marqueurs	29
2.2.2	Systèmes à capteurs actifs	30
2.2.3	Systèmes à capteurs passifs	31
2.2.4	Systèmes de reconstruction multi-vues	32
2.3	Classification des méthodes de reconstruction multi-vues	33
2.4	Reconstruction temporellement incohérente	34
2.4.1	Reconstruction basée silhouettes	34
2.4.2	<i>Space carving</i>	37
2.4.3	Reconstruction basée stéréo	37
2.4.4	Autres approches	39
2.4.5	Application à la reconstruction dynamique et limitations	39
2.5	Reconstruction temporellement cohérente	39
2.5.1	Recalage de formes dynamiques	40
2.5.2	Méthodes basées modèles non réalistes	41
2.5.3	Méthodes basées modèles réalistes	44
2.5.4	Méthodes libres temporellement cohérentes	46
2.6	Conclusion	48

2.6.1	Discussion	48
2.6.2	Présentation de notre approche	49
3	Extraction de mouvements	51
3.1	Introduction	55
3.2	Positionnement par rapport à l'état de l'art	55
3.2.1	Estimation de mouvements	55
3.2.2	Suivi de mouvements 3D	56
3.2.3	Notre approche	57
3.3	Description des données d'entrée	58
3.3.1	Séquences de volumes	59
3.3.2	Cartes de distances	59
3.4	Appariement de voxels	60
3.4.1	Critère de proximité	61
3.4.2	Critère d'orientation	61
3.4.3	Critère colorimétrique	61
3.5	Régularisation	62
3.6	Implémentation et optimisations	63
3.6.1	Limitation des appariements multiples	63
3.6.2	Prédiction	63
3.7	Résultats	64
3.7.1	Evaluation qualitative du champ de déplacements	65
3.7.2	Discussion des paramètres	66
3.7.3	Animation de maillage	67
3.7.4	Limitations	69
3.8	Conclusion	71
4	Animation de maillage	73
4.1	Introduction	77
4.2	Positionnement par rapport à l'état de l'art	77
4.2.1	Recalage 3D non rigide	79
4.3	Déformation pseudo-rigide	80
4.3.1	Concept	80
4.3.2	Coordonnées différentielles	80
4.3.3	Transformation <i>As-Rigid-As-Possible</i>	81
4.4	Processus d'animation	82
4.4.1	Maillage de référence	82
4.4.2	Sélection des points d'ancrage	82
4.4.3	Animation de maillage pseudo-rigide	84
4.5	Optimisation locale	89
4.5.1	Première approche : intégration numérique	89
4.5.2	Seconde approche : minimisation d'énergie	90
4.6	Résultats	92
4.7	Conclusion	93
5	Résultats	95
5.1	Introduction	99
5.2	Jeux de données	99
5.3	Présentation des résultats	99
5.3.1	Procédure et paramètres	100

5.3.2	Temps de calculs	101
5.3.3	Tests et discussion	101
5.3.4	Mesures du suivi de maillage	104
5.4	Applications	107
5.5	Limitations	109
5.6	Conclusion	110
6	Conclusion	111
6.1	Résumé	115
6.2	Travaux futurs	116
6.2.1	Améliorations à court terme	116
6.2.2	Poursuite des travaux à long terme	117
A	Publications et communications	135
B	Algorithmes	137
B.1	Transformée en distances Euclidiennes	137
B.2	<i>Poisson disk sampling</i>	139
C	Développements mathématiques	141
C.1	Discrétisation de Gaussienne	141
C.2	Table: loi centrée réduite	143
C.3	Flots optiques 3D	144
C.4	Estimation de transformation rigide	146
C.5	Matrice Laplacienne	147
C.6	Intégration numérique	148

Table of contents

1	Introduction	5
1.1	Context	9
1.2	Multi-view reconstruction	10
1.2.1	Outline	10
1.2.2	Reconstruction approaches	12
1.2.3	Usual techniques	12
1.3	RECOVER 3D	14
1.3.1	Objectives	14
1.3.2	Reconstruction process	16
1.4	Problem statement	18
1.5	Objectives	20
1.6	Input	21
1.6.1	Dataset	22
1.7	Contributions	22
1.8	Overview	24
2	Previous work	25
2.1	Introduction	29
2.2	Reconstruction systems	29
2.2.1	Marker-based systems	29
2.2.2	Active sensors system	30
2.2.3	Passive sensors system	31
2.2.4	Multi-view reconstruction system	32
2.3	Classification of multi-view reconstruction methods	33
2.4	Temporally inconsistent reconstruction	34
2.4.1	Silhouette-based reconstruction	34
2.4.2	Space carving	37
2.4.3	Stereo-based reconstruction	37

2.4.4	Other approaches	39
2.4.5	Application to dynamic reconstruction and limitations	39
2.5	Temporally consistent reconstruction	39
2.5.1	Dynamic shape registration	40
2.5.2	Non-realistic model-based methods and motion tracking	41
2.5.3	Realistic model-based methods	44
2.5.4	Temporally consistent model-free methods	46
2.6	Conclusion	48
2.6.1	Discussion	48
2.6.2	Our approach	49
3	Motion extraction	51
3.1	Introduction	55
3.2	Positioning in the context of previous work	55
3.2.1	Motion estimation	55
3.2.2	3D motion tracking	56
3.2.3	Our approach	57
3.3	Input sequence representation	58
3.3.1	Volume series	59
3.3.2	Distance volume coding	59
3.4	Voxel matching	60
3.4.1	Proximity criterion	61
3.4.2	Orientation criterion	61
3.4.3	Colorimetric criterion	61
3.5	Motion flow regularization	62
3.6	Implementation and improvements	63
3.6.1	Limitation of multiple matchings	63
3.6.2	Prediction	63
3.7	Results	64
3.7.1	Evaluation of the motion flow reconstruction	65
3.7.2	Discussion on the chosen parameters	66
3.7.3	Mesh animation	67
3.7.4	Limitations	69
3.8	Conclusion	71
4	Mesh animation	73
4.1	Introduction	77
4.2	Positioning in the context of previous work	77
4.2.1	3D non-rigid registration	79

4.3	Pseudo-rigid deformation	80
4.3.1	Concept	80
4.3.2	Differential coordinates	80
4.3.3	As-Rigid-As-Possible mesh processing	81
4.4	Animation process	82
4.4.1	Template mesh	82
4.4.2	Anchors' selection	82
4.4.3	Pseudo-rigid mesh animation	84
4.5	Local optimization	89
4.5.1	First approach: numerical integration	89
4.5.2	Second approach: energy minimization	90
4.6	Results	92
4.7	Conclusion	93
5	Results	95
5.1	Introduction	99
5.2	Datasets	99
5.3	Results	99
5.3.1	Process flow and parameters	100
5.3.2	Computing times	101
5.3.3	Experiments and discussion	101
5.3.4	Mesh tracking measurement	104
5.4	Applications	107
5.5	Limitations	109
5.6	Conclusion	110
6	Conclusion	111
6.1	Summary	115
6.2	Future work	116
6.2.1	Short-term	116
6.2.2	Long-term	117
	References	123
	A Publications and communications	135
	B Algorithms	137
B.1	Euclidean Distance Transform	137
B.2	Poisson disk sampling	139

C Mathematical details	141
C.1 Discretization of the Gaussian function	141
C.2 Table: reduced centered random variable	143
C.3 3D optical flows	144
C.4 Rigid motion computation	146
C.5 Laplacian matrix	147
C.6 Numerical integration	148
 Index	 149

List of figures

1.1	Temporal consistency issue in multi-view reconstruction	9
1.2	Multi-view reconstruction principle	11
1.3	Multi-view reconstruction example	12
1.4	Silhouette-based reconstruction	13
1.5	RECOVER 3D general pipeline	15
1.6	Architecture of the RECOVER 3D studio (Paris)	17
1.7	Architecture of the RECOVER 3D cyber-dome (La Réunion)	18
1.8	Different types of mesh sequences	19
1.9	Input data	22
2.1	Marker-based motion capture	29
2.2	Example of active sensor system: Kinect camera	30
2.3	Video-based reconstruction systems (passive sensors)	32
2.4	Multi-view reconstruction systems	33
2.5	Visual hull reconstruction	35
2.6	Visual hull reconstruction	36
2.7	Visual hull limitations	36
2.8	Binocular geometry for stereo-based reconstruction	37
2.9	Different types of multi-baseline geometries for multi-stereo capture	38
2.10	Examples of model-based methods [75, 17, 43]	42
2.11	Examples of model-based methods [164, 45, 61]	44
2.12	Examples of model-free methods [140, 91]	46
3.1	General process flow	58
3.2	Volume processing	59
3.3	Voxel matching between two consecutive volumes	60
3.4	Examples of inaccurate matching between two surfaces	61
3.5	Forward and backward matching between the two volumes. Gaussian filter applied to the raw vector fields and final motion field	62
3.6	Accumulated motion flows through several frames of five test sequences	64
3.7	Motion field regularization	65
3.8	Result of the two 3D optical flow computation between two frames of the <i>girl</i> sequence	66
3.9	Influence of the matching criteria	67
3.10	Overview of the mesh animation process	68

3.11	Evolution of the average Hausdorff distance during the <i>girl</i> and <i>short cheerleader</i> sequences	69
3.12	Representation of the texture matching metric on the animated mesh for the <i>short cheerleader</i> (a) and <i>dancer</i> (b) sequences	70
3.13	Stress case	70
4.1	3D registration	79
4.2	Laplacian vector on a triangle mesh	80
4.3	Inter-frame animation process	81
4.4	First volume of the sequence and associated template mesh	82
4.5	Anchor vertices selection	83
4.6	Pseudo-rigid mesh deformation	85
4.7	Silhouette fitting	90
4.8	Local optimization	91
4.9	Comparison between the two approaches of local optimization	92
4.10	Comparison between the two approaches of local optimization	93
5.1	Results of the mesh animation for the <i>cheerleader (short)</i> and <i>astronaut</i> sequences	102
5.2	Results of the mesh animation for the <i>dancer</i> and <i>capoeira</i> sequences	103
5.3	Example of free-moving clothes' tracking in the <i>dancer</i> sequence	103
5.4	Surface matching measurement for the <i>short cheerleader</i> , <i>astronaut</i> , <i>dancer</i> , and <i>capoeira</i> sequences	105
5.5	Surface matching measurement for the <i>long cheerleader</i> sequence	106
5.6	Capoeira sequence, comparison with de Aguiar <i>et al.</i> [45].	106
5.7	Comparison between the temporally inconsistent mesh sequence from a model-free reconstruction and our animated mesh	107
5.8	Applications	108
5.9	Applications	108
5.10	Mesh tracking limitations	109
5.11	Surface collapse during mesh tracking	109
5.12	Virtual crowd composed by several cloned <i>cheerleader</i> animations in a virtual scene	110
B.1	First step of EDT computation	138
B.2	Second step of EDT computation	138
B.3	EDT computation: (a)original binary volume, (b) external EDT, (c) combined EDT (internal + external).	138
B.4	Example of Poisson Disk sampling in a 2D space	140

List of tables

2.1	Taxonomy: overview of temporally consistent reconstruction approaches cited in this document.	40
3.1	Average motion flow computing times between two consecutive poses.	65
3.2	Mesh matching measurement (average Hausdorff distance) between animated mesh and visual hull of the target pose.	69
5.1	Datasets.	100
5.2	Average inter-frame computing time in seconds.	101

List of algorithms

1	Anchors' selection	84
2	Iterative solver for energy minimization.	87
3	Iterative solver - methods' details	88
4	Euclidean Distance Transform computation	137
5	Poisson disk vertex sampling	140

Introduction générale

Reconstruction multi-vues

Avec l'amélioration constante des technologies de synthèse d'images, la production de médias audiovisuels tend à mélanger de plus en plus les images issues de prises de vues réelles à celles générées par ordinateurs. Cependant, les techniques actuelles utilisées pour la composition de ces images hybrides restent limitées, notamment lors de la capture vidéo des acteurs que l'on souhaite placer dans un environnement virtuel. En effet, la prise de vues des personnages réels, effectuée en studio, et la modélisation de décors numériques sont réalisées séparément et doivent respecter des contraintes fortes (angles de vues, trajectoires des caméras, position des éléments de la scène ...) pour que ces différentes sources puissent être assemblées lors de la post-production. La solution alors envisagée pour contourner ces limitations consiste à réaliser une capture en trois dimensions de l'acteur de manière à pouvoir en réaliser un modèle numérique. Ce clone virtuel peut ensuite être utilisé, comme n'importe quel objet 3D numérique, lors de la modélisation et de l'animation d'une scène virtuelle. Cependant les méthodes actuelles, comme la capture de mouvements, ne sont pas adaptées à un tel usage. Ce type d'approche est plutôt destiné à capturer les gestes d'un acteur, *via* des méthodes le plus souvent invasives, de manière à les transposer sur un modèle numérique qui peut ainsi être animé de manière réaliste. Dans ce contexte, les technologies de reconstruction multi-vues présentent des propriétés intéressantes. Ce type d'approche consiste à capturer la surface d'un objet à partir d'acquisitions vidéo non-invasives. Ces acquisitions sont réalisées par un ensemble de caméras disposées à de multiples positions autour de la scène à reconstruire. Cette approche est accessible à des infrastructures réduites car elle emploie un matériel standard (caméras vidéo uniquement) et peut être réalisée dans des studios de taille réduite (les dimensions de la zone de capture sont limitées à la performance de l'acteur). Appliquées sur des acteurs, cette méthode permet de produire un objet 3D qui représente l'acteur tel qu'il a été filmé en studio, avec costume et accessoire. Cet objet 3D peut ensuite être exporté vers un environnement virtuel et être manipulé à l'aide de logiciel d'animation 3D. A ce stade, le rendu final est effectué à partir d'une scène entièrement virtuelle. La prise de vues peut alors être effectuée avec une caméra virtuelle qui n'est plus limitée aux contraintes physiques de la capture de l'acteur en studio et peut de cette manière adopter des points de vue et des mouvements totalement libres dans l'espace de la scène virtuelle. La représentation de l'acteur sous forme de modèle 3D permet également d'appliquer diverses opérations telles que le clonage des personnages ou le ré-éclairage par des sources lumineuses virtuelles.

RECOVER 3D

Le projet RECOVER 3D (*Real-time Environment for COmputational Video Editing and Rendering in 3D*), financé par le programme *Investissements d'Avenir* dans le cadre du *Fonds national pour la Société Numérique* (FSN), est issu d'une collaboration entre différents acteurs industriels et académiques. L'objectif de ce projet est de concevoir et développer une chaîne de production complète pour adapter la reconstruction multi-vues à la production télévisuelle, de la captation vidéo jusqu'à la diffusion (voire Figure 1.5). le consortium RECOVER 3D est formé de quatre partenaires :

- L'entreprise XD Productions est le leader du projet. Cette société basée à Issy-les-Moulineaux (Paris) héberge le prototype du studio de capture vidéo multi-points de vues et assume la direction technique.
- Le CReSTIC (Centre de Recherche en STIC) de l'Université de Reims Champagne-Ardenne assure la direction scientifique du projet. L'équipe SIC (Signal, Image et Connaissance) a lancée deux sujets de thèse (dont celle décrite dans ce manuscrit) portant sur l'amélioration des technologies de reconstruction multi-vues.
- Le groupe Euro Media apporte une expertise technique pour l'étape de diffusion des résultats de la reconstruction. Cette entreprise a mis en place une solution logicielle pour la manipulation des images issues de l'hybridation des décors virtuels et d'acteur reconstruits.
- L'ILOI (Institut de l'Image de l'Océan Indien) est une école de production multimédia basée à la Réunion. Elle accueille une seconde version du studio de reconstruction multi-vues qui doit permettre la validation des technologies RECOVER 3D à échelle réelle.

La technologie RECOVER 3D est basée sur un studio de capture où un ensemble de caméras (24 pour le studio parisien) est disposé autour de la zone de capture où se tient l'acteur. Ces caméras sont réparties en blocs *monoscopiques* (une seule caméra) et *multiscopiques* (4 caméras alignées côte à côte). La méthode de reconstruction actuelle suit une approche *basée silhouettes* : un masque correspondant à la silhouette de l'acteur est extrait dans chaque image acquise simultanément par l'ensemble des blocs monoscopiques grâce à un algorithme de *chroma-keying*. Pour chaque pixel du contour d'une silhouette, un rayon peut être lancé à partir du centre optique de la caméra correspondante vers ce pixel, ce qui permet d'obtenir une demi-droite sur laquelle se trouvent tous les points de l'espace qui peuvent potentiellement correspondre à ce pixel. En répétant l'opération pour chaque pixel du contour de la silhouette, l'ensemble des demi-droites obtenues forme un *cône de silhouette*. L'intersection des cônes de silhouettes obtenus à partir de chaque point de vue permet de calculer l'*enveloppe visuelle* de l'acteur. Cette reconstruction est opérée grâce à un algorithme volumique : l'espace de la zone de capture est discrétisé pour former une grille 3D dont chaque élément, nommé *voxel* (*Volume Element*), est associé à une valeur binaire (*intérieur* ou *extérieur* à la forme reconstruite). Ce volume discret est *creusé* par les cônes de silhouettes (les voxels qui ne se trouvent pas à l'intérieur du cône sont marqué *extérieurs*) jusqu'à ne conserver que le volume de l'enveloppe visuelle. Le résultat de cette reconstruction est donc un volume qui représente la forme de l'acteur. Ce type de reconstruction a l'avantage d'être rapide et peu complexe. Il présente cependant des inconvénients, notamment son incapacité à reconstruire les concavités de la surface. Pour améliorer le résultat de cette reconstruction, des travaux de recherches sont menés dans le cadre du second sujet de thèse RECOVER 3D. L'objectif est d'utiliser les blocs de caméras multiscopiques pour effectuer une reconstruction par *stéréo-matching* et de la fusionner avec l'enveloppe visuelle pour obtenir une reconstruction plus précise et de

meilleure qualité. Les travaux décrits dans ce manuscrit sont cependant basés uniquement sur des reconstructions basées silhouette.

Problématique

L'inconvénient des méthodes de reconstruction par enveloppe visuelle ou par *stéréomatching* est leur incapacité à représenter une forme dynamique évoluant au cours du temps. En effet ces méthodes sont initialement utilisées pour reconstruire des objets statiques à partir d'un ensemble de photos multi-points de vues. Pour reconstruire la performance d'un acteur, on utilise un ensemble de caméras synchronisées. A chaque pas de temps de l'acquisition, on dispose ainsi d'une image pour chaque point de vue, ce qui permet d'effectuer la reconstruction. Ce processus est répété tout au long des séquences vidéo de la capture. Le résultat est une série d'objets 3D statiques qui correspondent aux poses successives de l'acteur au cours de l'acquisition vidéo. Pour effectuer le rendu de cette reconstruction, chaque pose de cette séquence doit être affichée à la *frame* (image) correspondante. Ce mode de fonctionnement n'est pas adapté aux logiciels d'animation et empêche toute interaction entre le clone numérique du personnage et son environnement virtuel. L'objectif des travaux de cette thèse est de transformer ces séquences d'objets 3D statiques en un unique modèle animé. Le résultat souhaité est un maillage animé et temporellement cohérent. La forme de l'acteur est définie par un maillage triangulé dont la connectivité et la topologie restent constantes au cours de l'animation : seule la position des sommets évolue au cours du temps. De plus, notre méthode doit être adaptée à tous types de silhouette. En effet, la technologie RECOVER 3D est amenée à effectuer la reconstruction d'acteurs dans différentes situations et filmés en conditions de tournage réelles, c'est à dire en costume de scène. Les silhouettes ainsi capturées ne correspondent pas toujours à des morphologies humaines (ensemble de membres articulés), notamment à cause de l'utilisation de vêtements amples, de robes ou d'accessoires. Notre méthode ne pourra donc pas être basée sur les modèles articulés habituellement utilisés pour la reconstruction de mouvements humains, mais devra au contraire supporter les mouvements de surface de types *libres*. L'approche décrite dans ce manuscrit est basée sur deux étapes principales : l'extraction des mouvements de l'acteur puis la déformation d'un modèle de maillage selon ces mouvements pour finalement obtenir un personnage animé dont l'évolution au cours du temps suit les poses décrites par la reconstruction initiale. Ce manuscrit est organisé en cinq chapitres :

- Le chapitre 1 présente en guise d'introduction le contexte du projet RECOVER 3D et les contraintes auxquelles sont soumis nos travaux.
- Le chapitre 2 propose une revue exhaustive des différentes approches de l'état de l'art, usuelles ou expérimentales, existant actuellement pour la reconstruction de formes dynamiques et en particulier les performances d'acteurs.
- Le chapitre 3 détaille la méthode que nous avons développée pour extraire un flot de mouvements qui définit les gestes de l'acteur au cours de la capture.
- Le chapitre 4 décrit ensuite notre méthode de déformation de maillage guidée par le flot de mouvements et basée sur des approches *pseudo-rigides*.
- Le chapitre 5 présente les résultats de l'ensemble de notre méthode et la manière dont ils peuvent être utilisés dans le cadre d'une chaîne de production télévisuelle.
- Le chapitre 6 conclue ce manuscrit en faisant le bilan de nos travaux et en proposant des pistes pour poursuivre ces recherches.

Chapter 1

Introduction

Contents

1.1	Context	9
1.2	Multi-view reconstruction	10
1.2.1	Outline	10
1.2.2	Reconstruction approaches	12
1.2.3	Usual techniques	12
1.3	RECOVER 3D	14
1.3.1	Objectives	14
1.3.2	Reconstruction process	16
1.4	Problem statement	18
1.5	Objectives	20
1.6	Input	21
1.6.1	Dataset	22
1.7	Contributions	22
1.8	Overview	24

Résumé

Ce chapitre d'introduction présente le contexte de nos travaux de recherche ainsi que la problématique de ce sujet de thèse. Dans un premier temps, nous donnons un aperçu de la technologie de reconstruction multi-vues ainsi que de son usage dans le cadre de la production audiovisuelle (*cf.* Section 1.1). Cette technique permet d'effectuer une reconstruction de la surface d'un objet à partir d'un ensemble d'images prises de différents points de vue répartis autour de la zone de capture. Les avantages de cette technologie sont d'utiliser uniquement des caméras vidéo et d'être non invasive. Appliquée à la reconstruction d'acteurs, elle permet de capturer la performance d'un personnage filmé en studio et de le reproduire sous la forme d'un objet 3D virtuel.

Les méthodes de reconstruction multi-vues les plus répandues sont introduites dans la Section 1.2. Nous décrivons brièvement les approches basées silhouettes et celles basées stéréo (*cf.* Section 1.2.3). Ces techniques utilisent un ensemble d'images statiques pour effectuer la reconstruction d'un objet. Dans le contexte de la production de contenu animé, on utilise des caméras synchronisées pour filmer un acteur en mouvement. On peut alors extraire un ensemble d'images pour chaque *frame* des séquences vidéos et effectuer une reconstruction statique au pas de temps correspondant. Une revue plus détaillée de ces différentes approches est présentée au Chapitre 2 (*cf.* Sections 2.4.1 à 2.4.3).

Nous présentons ensuite le projet RECOVER 3D (*cf.* Section 1.3) dans le cadre duquel se sont déroulés les travaux de recherche décrits dans ce manuscrit. Ce projet est issu d'une collaboration entre différents partenaires industriels et académiques. Son objectif est d'adapter les technologies de reconstruction multi-vues aux contraintes de la production télévisuelle. Un procédé de reconstruction innovant a été développé, basé sur une approche hybride mélangeant les approches silhouettes et stéréo. Cette reconstruction est adaptée aux studios *chroma-key* de télévision. Le projet RECOVER 3D prévoit également la mise en place d'un pipeline de production complet autour de cette méthode de reconstruction, allant de la capture studio à la diffusion de contenus hybrides mélangeant clones numériques d'acteurs et environnements virtuels.

La problématique identifiée au cours de cette thèse (*cf.* Section 1.4) concerne la cohérence temporelle des reconstructions. En effet, les méthodes utilisées permettent de calculer une reconstruction indépendante pour chaque *frame* des captures vidéos. Le résultat est une succession de poses statiques sous la forme d'objets 3D dépourvus de continuité structurelle: chaque pose est un objet indépendant sans correspondance établie avec les autres poses de la séquence. Ce mode de représentation n'est pas adapté aux traitements de post-production et présente de nombreux inconvénients lors de l'utilisation des acteurs reconstruits par des logiciels d'animation 3D (*cf.* Section 1.5). Nous décrivons ensuite les données que nous avons utilisées lors de nos travaux et la manière dont elles sont générées (*cf.* Section 1.6). Une description plus exhaustive de chaque jeux de données testé est disponible au Chapitre 5 (*cf.* Section 5.2). Nous détaillons dans la Section 1.7 les contributions apportées par cette thèse. Enfin, la Section 1.8 annonce le plan de ce manuscrit.

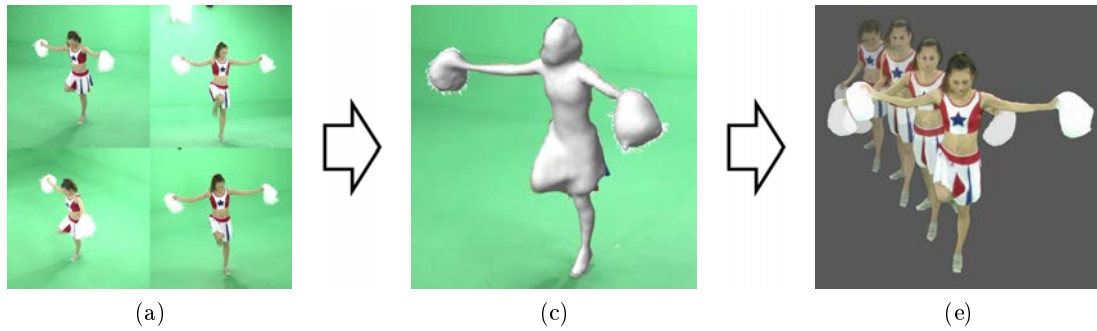


FIGURE 1.1 – **Temporal consistency issue in multi-view reconstruction:** From a markerless space-time capture of an actor (a), we automatically construct a 3D reconstruction (b) for each frame, which produces time-series of *snapshot* poses (c). Our goal is to add temporal consistency to these dynamic reconstructions.

1.1 Context

The *multi-view reconstruction* technologies provide a mean to transform an actor performance, shot from multiple viewpoints by a set of video cameras into an animated object which represents the actor as a 3D time-varying model. This technique links traditional video capture to new technologies of computer generated pictures. The reconstruction of a scene and its evolution throughout time has several applications such as media content production, human-machine interaction, motion analysis or virtual reality. The context of our research is the media production. Due to the multiplication of channels and the appearance of new consumption behaviors (VOD, Internet ...), the TV audience is increasingly fragmented. Broadcasters and producers seek differentiated quality contents, produced in optimal economic conditions. The use of 4D reconstruction studios is becoming a new paradigm to perform the acquisition of a 3D scene and convert it into a digital representation, suited to a 3D animation framework. These indoor infrastructures provide controlled environments generally based around a large room with uniform background equipped with multiple synchronized calibrated video cameras and appropriate illumination.

The current evolution of actor's performance capture for media production is to avoid the use of constraining and invasive components such as marker-based technologies. If the capture of human motion is widely used for realistic animation, the recent trend is to digitize the actor him/herself to enable the production of hybrid computer-generated images. The actor is filmed in a studio, in the same conditions as for a traditional indoor shooting. The key idea is to perform a set of 2D video captures that will be used to compute a three-dimensional representation of the actor. This reconstruction will be natively adapted to the production of virtual scenes in computer-generated environments, instead of the complex compositing operations currently applied.

The 3D reconstruction of static objects from a set of 2D pictures has already been widely studied but the capture of animated contents is a challenging task that needs to include a dynamic information into the usual models. The most intuitive reconstruction approaches for an actor's performances can be described as a 3D extension of regular video capture. They compute a static reconstruction of the scene's content at regular time steps (which correspond to each frame of the shooting), such as a 2D video is a sequence (or

time series) of snapshot pictures (see Figure 1.1). The animation is then figured by successively rendering these poses, just as the frames of a video where the motion information is not captured but simply interpreted by the human brain. However, this representation is not well-suited for computer-animated production. The use of reconstructed actors in media content needs a dynamic description of the scene. The reconstructed subject must be structured as a 3D animated character to be included in a 3D modeling framework. Usually, these systems exploit the high redundancy of video sequences while leveraging the assumed continuity of movement. A review of recent trends in multi-view dynamic scenes' reconstruction from videos identifies two major building blocks required to, first, reconstruct 3D volumetric or surface data and second, track feature points over time.

1.2 Multi-view reconstruction

We describe in this part the concept of multi-view reconstruction (see Section 1.2.1 and the two main families of approaches to achieve it (see Section 1.2.2). We then introduce the most usual techniques used in our context in Section 1.2.3.

1.2.1 Outline

As the computer generated images are increasingly used in the media industry, the composition of pictures from various sources is a usual operation: cinema and video games increasingly combine real images with computer-generated contents. More particularly, real actors are commonly included in virtual environments. Technologies such as *matte painting* allow to mix actor's appearances with digital backgrounds. This technique is massively used by the cinema industry and also becomes more and more present in TV production as it is an easy way to place the characters, shot in an indoor *chroma-key* studio, in any type of scene. Chroma-keying is a video acquisition which takes place against a uniform *key color* background (usually green or blue). Unfortunately, this compositing technology is hardly constrained by the studio facilities. The camera viewpoint and motion for the rendering of the final composed scene are limited by the degree of freedom of the real cameras which shoot the actors, and by the dimensions of the studio. If most of the cinema production teams get larger infrastructures which limit these problems, this is a critical drawback for TV industry, where the production time and budget are reduced.

The *motion capture* (*MoCap*) is another common technology that registers the motions of an actor and transfer them to a virtual character, leading to a realistic animation. The registered motions are captured following a set of predefined key points which correspond to the articulations of the virtual model. This model is then animated with a constraining structure, most of the time a skeleton, which is itself driven by the captured motions. The main limitation of this system is the lack of genericity: the articulated 3D model have to be adapted to the motion capture system. Most of the MoCap systems use a set of physical markers which have to be put on the performing actor, requiring an onerous pre-production's manual step. The motions of these markers are then captured by a set of multi-viewpoint cameras, usually infra-red cameras. Nevertheless, other systems perform marker-free MoCap. Recently, new miniaturized systems, such as *Kinect* cameras (see Section 2.2), bring marker-free MoCap to small infrastructures. However, these systems are limited in the captured degrees of freedom and lack of precisions compared to traditional

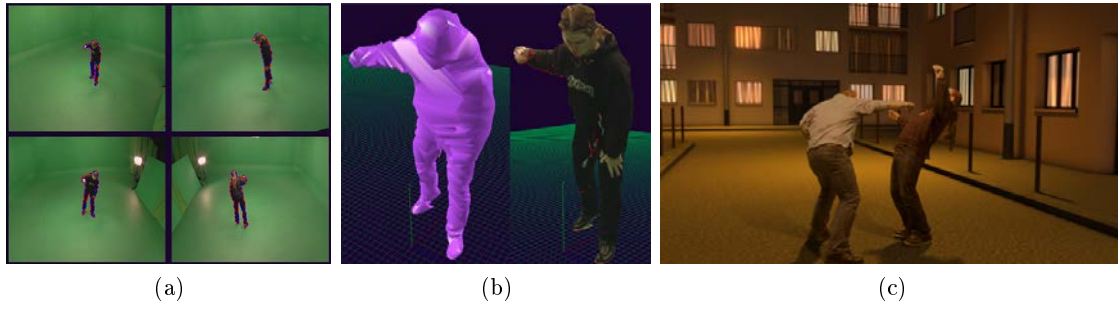


FIGURE 1.2 – **Multi-view reconstruction principle:** (a) multi-viewpoint video sequences, (b) reconstructed surface, (c) digital clones used as 3D characters in a computer generated scene (source: XD Productions).

systems. But the main problem of MoCap technologies is that the shooting conditions are heavily constrained as the actors have to wear adapted suits. The animated model then represents a virtual character whereas our goal is to get a representation of the actor him/herself. However, the output is an animated 3D model which is then inserted into a virtual environment. This virtual scene presents the advantages we are looking for. The rendering is performed by a virtual camera which has no constraints on viewpoints and trajectories. The industrial use of motion capture requires a real-time visualization of animations on shooting location. This allows directors to guide their actors accurately. In the case of scenes which contain, for instance, virtual crowds or furnitures, the actors can have a video feedback allowing them to better understand the environment they are supposed to be in, and adapt their performance accordingly.

As an alternative to these technologies, the multi-view reconstruction uses a set of multi-viewpoint cameras, shooting an indoor studio performance of an actor, to reconstruct a 3D representation of the character. The core idea is to mix the captured performance and computer-generated images earlier in the production process, simultaneously to the studio shooting. This way, producers can use a three-dimensional result to judge whether a scene will appear in the final production and to guide actors as well allowing the performance to be directly inserted into a traditional computer graphics production pipeline. The actors perform with costumes and accessories and their *virtual clone* will then be transposed in a virtual environment, as depicted in Figure 1.2. The advantage of this technique, compared to simple chroma-keying, is that the virtual clone is transposed in the 3D space, as a usual animated character. The rendering of the scene is then processed by a virtual camera in the 3D scene. This unconstrained camera can move freely, allowing for example unrealistic viewpoints or large plans. Another important advantage is that the virtual character can be duplicated, thus creating crowds from a small dataset of reconstructed actors' performances (see Figure 1.3). Finally, this multi-view reconstruction can be performed in a reduced studio and therefore offers complex compositing operations to low-budget productions. Using multi-view reconstruction filming sets, seeking to reiterate what has been done for motion capture, provides the same facilities to teams using lighter and generic infrastructures. Section 1.2.2 and 1.2.3 of this introduction present a quick overview of all these technologies. Details are given in Chapter 2. For in-depth studies of this wide research area, one can refer to several comprehensive books [134, 107, 111, 129, 104].



FIGURE 1.3 – **Multi-view reconstruction example:** A crowd scene released with the reconstruction of a reduced set of actors, filmed independently in a TV studio, wearing their costume. The virtual clones are then duplicated and shifted in space and time. The scene is rendered with a computer-generated environment and a virtual camera (source: XD Productions).

1.2.2 Reconstruction approaches

Multi-view reconstruction approaches proposed until now can be split into two families of methods:

- *Model-based* approaches use a reference geometric description (for example a triangular mesh) of the character. This prior knowledge can be manually constructed as a generic human body or obtained using an acquisition system (a 3D scanner for instance). This input model is then animated in accordance with the motions of the actor.
- *Model-free* methods that can be distinguished by the fact that no prior knowledge (relating to the nature and number of objects, characters' morphologies) is provided to the system. The vast majority of these techniques rely on a reconstruction of the scene's content for each frame of the videos, independently from each other's. Due to their general nature, these techniques often generate results without any temporal coherence.

Chapter 2 presents a survey on these approaches (see Sections 2.4 and 2.4).

1.2.3 Usual techniques

This section gives a brief overview of the two most common methods of multi-view reconstruction, which both belong to model-free category. More details on these techniques are given in Chapter 2 (see Sections 2.4.1 and 2.4.3).

1.2.3.1 Visual Hull

The *shape-from-silhouette* (or *silhouette-based*) reconstruction approaches use the actor's contours, filmed from several camera pictures around the scene, to compute a 3D

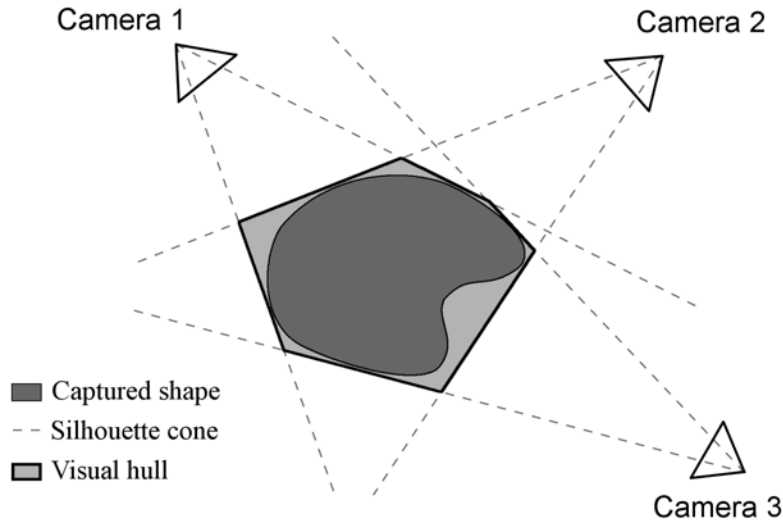


FIGURE 1.4 – **Silhouette-based reconstruction:** the visual hull of the shape is defined by the silhouettes extracted in the images of the cameras.

shape. This object, named *visual hull*, is a convex hull which contains the maximum volumes filled by the objects in the scene. The projection of this shape in the pictures matches the silhouettes. A silhouette is a binary mask associated with a given perspective which includes all pixels corresponding to the projection of a point of the 3D object to be reconstructed. A set of multiple cameras, located at various viewpoints, generate a multi-view pictures dataset. A mask of the actors is then extracted in each of these pictures. A silhouette's pixel (*i.e.*, which is lying on the contours of the actor's mask) and the optical center of the corresponding camera are linked by a half-line. This half-line contains all the points in the 3D scene which projects to this pixel, as shown by Figure 1.4. The set of all half-lines, corresponding to every pixel from the silhouette, belongs to a surface referred to as the *silhouette cone*. The visual hull, as defined by Laurentini [83], is the intersection of the silhouette cones associated to all the cameras, as presented in Figure 1.4. Silhouette-based reconstruction therefore involves estimating the visual hull of the 3D object, represented by a polygon in Figure 1.4. Several approaches can be used to estimate this shape. The *volumetric methods* compute the volume contained in this hull by successively carving the whole scene's volume with the silhouettes' projection in the 3D space and thus produce a discretized volume. The polyhedral (or *surface-based*) approaches directly compute the visual hull's surface by intersecting the silhouette cones, leading to a 3D mesh.

1.2.3.2 Stereo matching

The *shape-from-stereo* (or *stereo-based*) reconstruction approaches use several images taken from different cameras (or multi-sensors cameras, *e.g.*, *ZED camera*¹) to estimate the captured object. For each pixel of an image, we search its corresponding pixels in the other images using *correlation* criteria. This correlation should establish a correspondence between pixels from different images that correspond to the same 3D point of the captured scene. After finding the matching pixels, a *triangulation* process is applied to compute the 3D position of the points in the captured scene. Using a *simplified epipolar* geometry (see

1. <https://www.stereolabs.com/zed/specs>

Section 2.4.3), we obtain for each point a *disparity* value which is defined as the abscisse difference between matching pixels. As the disparity of a pixel is related to the depth of the 3D point [63], the triangulation step is significantly simplified and a *depth-map* is directly constructed from the disparities (see [147, 44] for details on correspondences and 3D reconstruction). A depth-map is an image which describes the distance of a point from a viewpoint. Each pixel is associated to a value (which can be represented as a grey level) that represents the depth of the 3D point, whose projection correspond to this pixel, from the camera viewpoint. In the case of circumposing acquisition system, composed of multiple cameras located around the scene, a *multi-stereovision* (*i.e.*, generalization of binocular stereovision) method proceeds by performing a stereo-based reconstruction from various viewpoints. In order to obtain a single reconstructed object, the acquired depth-maps are merged in a single point-cloud thanks to a rigid 3D registration algorithm such as *Iterative Closest Point* (ICP) [21, 37] (see Section 2.2.2).



1.3 RECOVER 3D

This thesis took place in the RECOVER 3D project (Real-time Environment for Computational Video Editing and Rendering in 3D), funded by the *Investissements d'Avenir* program. This project was carried out from 23rd January 2012 to 11th May 2015. Its main objective was to elaborate an integrated virtual video system for the motion picture market.

1.3.1 Objectives

The general goal of the project is to develop an innovative multi-view reconstruction method and to integrate it in an adapted production pipeline, from studio shooting to broadcasting. The innovation brought by RECOVER 3D aims at freeing the creation of video images from common material constraints linked to multi-cameras shooting. The multi-view reconstruction technologies will be used to create a new *virtual cloning* system of actors and scenes based on smart 3D video capture, natively delivering depth information. An indoor multi-view reconstruction studio (*cyber-dome*), driven by the RECOVER 3D software solutions, generates the digital transcription of the scene in three dimensions, following a multi-view stereo-based visual hull paradigm.

RECOVER 3D is a consortium based on a partnership between academic researchers in computer vision, industrial integrators, and producers from the broadcast world.

-  XD Productions provides animated content and motion capture technologies. This company, based in Issy-les-Moulineaux (Paris), hosts the prototype of the RECOVER 3D multi-view studio. XD Productions is the leader of the project and assumes the technical direction.
-  CReSTIC (Centre de Recherche en STIC) is a laboratory of the University of Reims Champagne-Ardenne. The SIC (Signal, Image et Connaissance) research team activities are focused on modeling, signal and images processing, and artificial

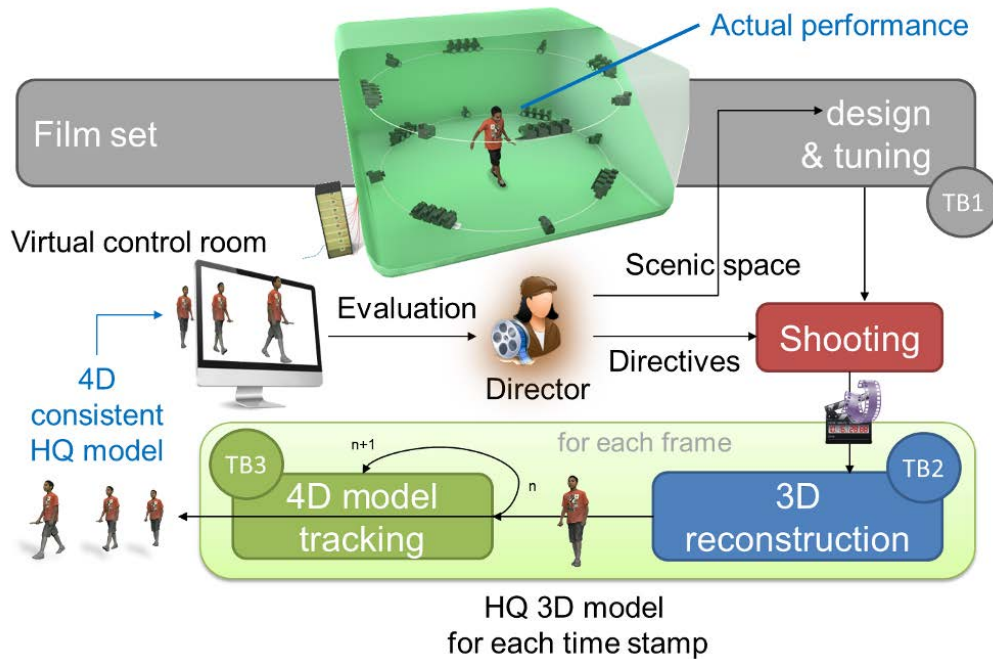




FIGURE 1.5 – RECOVER 3D general pipeline.

intelligence. It takes on the scientific direction. Two PhD theses, including the one described in this manuscript, have been carried out, as parts of this project.

-  Euro Media Group provides production and broadcasting facilities, as well as technical expertise. As an industrial partner in this project, it provided a technological integration of the RECOVER 3D software solution (*virtual control room*, described below).
-  ILOI (Institut de l'Image de l'Océan Indien) is a vocational school, located at Réunion, specialized in images and media production. ILOI is part of the ETNA network (*European Training Network for Animation*). A second version of the cyberdome has been installed in the ILOI studios (one of the largest virtual studio in France) to perform a full-scale validation of the RECOVER 3D technologies.

Capturing a real scene to generate its virtual representation is a challenging goal that led to many research work. There is currently no mainstream system that offers the possibility to reproduce a real actor's performance as a 3D, time-evolving, scene. The only solutions available for TV and cinema industries are based on stereoscopy and are therefore limited as they cannot reproduce the real volumes of the actors and objects. The RECOVER 3D objectives consist in bringing the third dimension to video industry thanks to innovative treatments on digital pictures and 3D objects. The constraint is not only to improve the overall esthetical quality of the resulting models, but also to produce them in real time or in reduced post processing delay, providing a credible alternative to standard 2D studios.

The RECOVER 3D process flow is composed of several steps, described in Figure 1.5. The first step is the indoor shooting of the actor by a set of multi-viewpoint cameras. These cameras are distributed on two rings around the scene (the first one on the floor and the second one at the ceiling) and alternates *monoscopic* and *multiscopic* blocks of one or four cameras, respectively. Next, the multi-viewpoint videos are used to compute the 3D reconstruction using an innovative approach where the usual silhouette-based method is enhanced with a set of depth-maps computed from each multiscopic viewpoint. The multi-view reconstruction can be performed either online or offline:

- *Online Mode*: the online mode enables the director to control the recording by a real-time reconstruction of the actor at 25 fps, used as a pre-visualization tool.
- *Offline Mode*: in the post-processing step, a higher quality reconstruction can be computed once the data were stored on a server. This process may require several minutes for each frame. The final result is a time series of textured meshes. These sequences can then be used in 3D animation software thanks to *ad hoc* plugins.

The work described in this manuscript takes place in this offline process, as an improvement of the model-free reconstruction previously described. The goal is to transform the reconstructed pose sequences in a temporally consistent model to obtain an animated virtual clone of the actor as final result. A *virtual control room* contains all the functionalities allowing visualization and interactions with the reconstructed sequences for the TV broadcasting, live or pre-recorded. It also supplies several tools to the directors for various operations (which can be applied live during the acquisition in online mode):

- Integration of one or several reconstructed models in the software.
- Addition of a virtual scenery.
- Real-time rendering of the scene. This functionality is required for live broadcasting and is also useful for pre-recorded programs where decisions must be quickly taken about set setting and composition.
- Handling virtual cameras in the scene. In particular: adjusting camera position and zoom.
- Perform camera motions (travelling, panoramic). The advantages of reconstruction technologies are highlighted here: with virtual cameras and reconstructed 3D models, any kind of trajectories is available without machinery additional cost (no crane shot or camera dolly, lower labor force need).
- Allowing to switch to stereoscopic virtual cameras, in order to produce 3D videos. The third dimension of the reconstructed models enables this operation from any viewpoint, making stereoscopic 3D rendering natural.

1.3.2 Reconstruction process

The RECOVER 3D pipeline is based on the multi-view reconstruction system developed by XD Production prior to the project starting date. This system contains a chroma-key studio and a set of multi-viewpoint cameras which perform a model-free reconstruction. This reconstruction uses a volumetric silhouette-based algorithm (see Section 2.4.1). The 3D scene is first discretized in a 3D voxel grid. The visual hull is carved in this grid, by projecting the silhouette mask from each camera, leading to a volumetric representation of the actor. The surface of this volume is then extracted to obtain a 3D mesh, which is finally textured by projecting the pixels from the multi-view images. This process is repeated at each frame of the videos, leading to a time series of static 3D meshes corresponding to the actor's pose at each frame. The goal of this reconstruction is to export the digital clone of

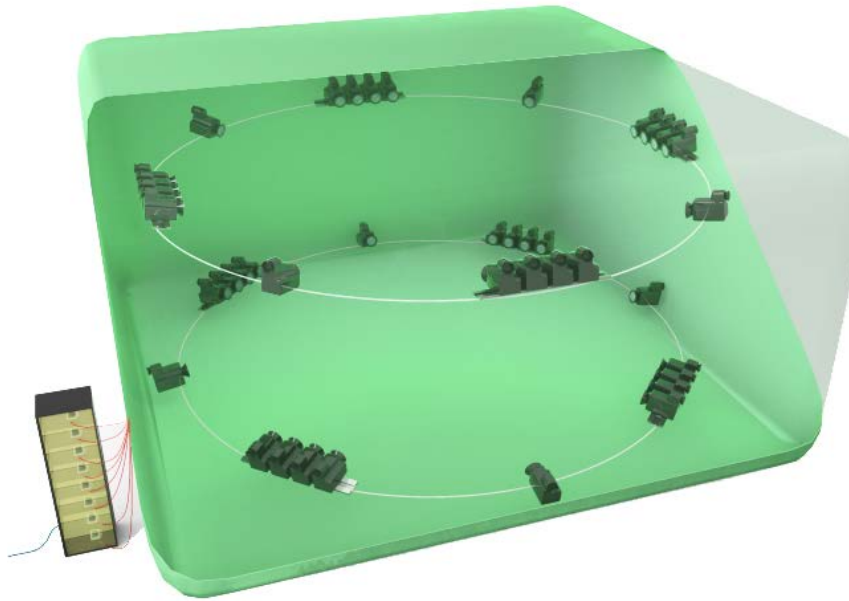


FIGURE 1.6 – Architecture of the RECOVER 3D studio. The videos cameras are distributed on two rings (floor and ceiling) around the capture area and alternate monoscopic units with multiscopic units.

the actor in a virtual environment. The quality of the results and the computing time both depends on the resolution of the voxel grid. However, advantages of the volumetric visual hull reconstruction are the low complexity and computing time. The captured animation is thus constructed by successively rendering the meshes of the sequence, with the same frame rate as the videos. The multi-view pictures are used for the texture mapping of the 3D meshes. A texture patchwork is created, where each triangle corresponds to a face of the mesh. For each point of this face, the normal vector is interpolated from the vertices' normals. This vector is then used to select the best oriented cameras from which the color of the pixel is deduced. This color is obtained by projecting the point toward the cameras to get the corresponding pixels which are blended to give the final color of the texel. This system already allows to performs a real-time reconstruction, with a 128^3 voxel resolution. Thus, a previsualization of the reconstructed actor in his/her virtual environment is available live, during the shooting of the scene. The actor can also have a live feedback of his/her performance, seeing the live reconstruction on a screen. The offline mode enables a better reconstruction, with a voxel grid resolution up to 512^3 . The meshes obtained after the surface extraction are simplified *via* a mesh decimation algorithm.

RECOVER 3D innovations The RECOVER 3D project takes place in the general framework of VBR (Video-Based Rendering) video processing. The originality of the proposed approach lies into the use of real-time reconstruction techniques. This approach named SBVH (Stereo-Based Visual Hull) consists in a combination of stereo-based and silhouette-based solutions. Concerning the stereo-based part, the project rely upon an existing pipeline which hardware and software architecture have been previously developed by the CReSTIC. This approach is based on multiscopic shooting, using a set of parallel axis cameras. The reconstruction system developed for the RECOVER 3D project uses several stereoscopic blocks that supply a set of multi-viewpoint depth-maps which are



FIGURE 1.7 – Architecture of the RECOVER 3D cyber-dome installed at ILOI (La Réunion).

be used to enhance the silhouette-based reconstruction, leading to high-quality models. The resulting 3D objects can be exploited afterwards in an innovative software system for the broadcasting of the final composed videos.

A new studio architecture was developed for the RECOVER 3D project. This new system distributes the viewpoints around the capture area, mixing multi-stereoscopic blocks and monoscopic cameras, as presented in Figure 1.6. Two prototypes were installed in the studios of XD Productions and ILOI.

- The **Paris studio** (XD Productions) is made of 24 full HD cameras (1920×1080 pixels) in a square room of $100m^2$ and 4.5m high. This studio is dedicated to experimental shooting. The datasets used in this thesis were produced in this studio.
- The **Réunion studio** (ILOI) is a cylinder with a 15-meter diameter and 6-meter height as shown in Figure 1.7. This cyber-dome is dedicated to the production of commercial content and is used for the validation of the project methods.

The CReSTIC research contributes to two main tasks, centered on two PhD thesis. The first task is to develop a new hybrid model-free approach, using both visual hull and shape-from-stereo, leading to a high quality model-free reconstruction (see Figure 1.5 TB2). The second task, described in this manuscript, aims at computing a temporally consistent animation from the reconstructed sequences (see Figure 1.5 TB3), as described in Section 1.4.

1.4 Problem statement

The reconstruction of a moving actor transforms a set of multi-view videos into a 3D time-evolving shape. This reconstruction is represented as a sequence of triangular meshes (with the same frequency as the video source) and their associated textures. Since a sequence of meshes can take a variety of forms, Arcila [15] proposed a formalism for describing the different types of time-varying meshes and identified the following categories: dynamic meshes, stable mesh sequences and unconstrained mesh sequences (see Figure 1.8). These distinctions are based on the existence of temporal coherence in meshes, both at topological and structural level:

- *Dynamic mesh*: number of vertices, connectivity (neighbourhood relationships between vertices) and topology stay unchanged during the sequence. It can be seen as a

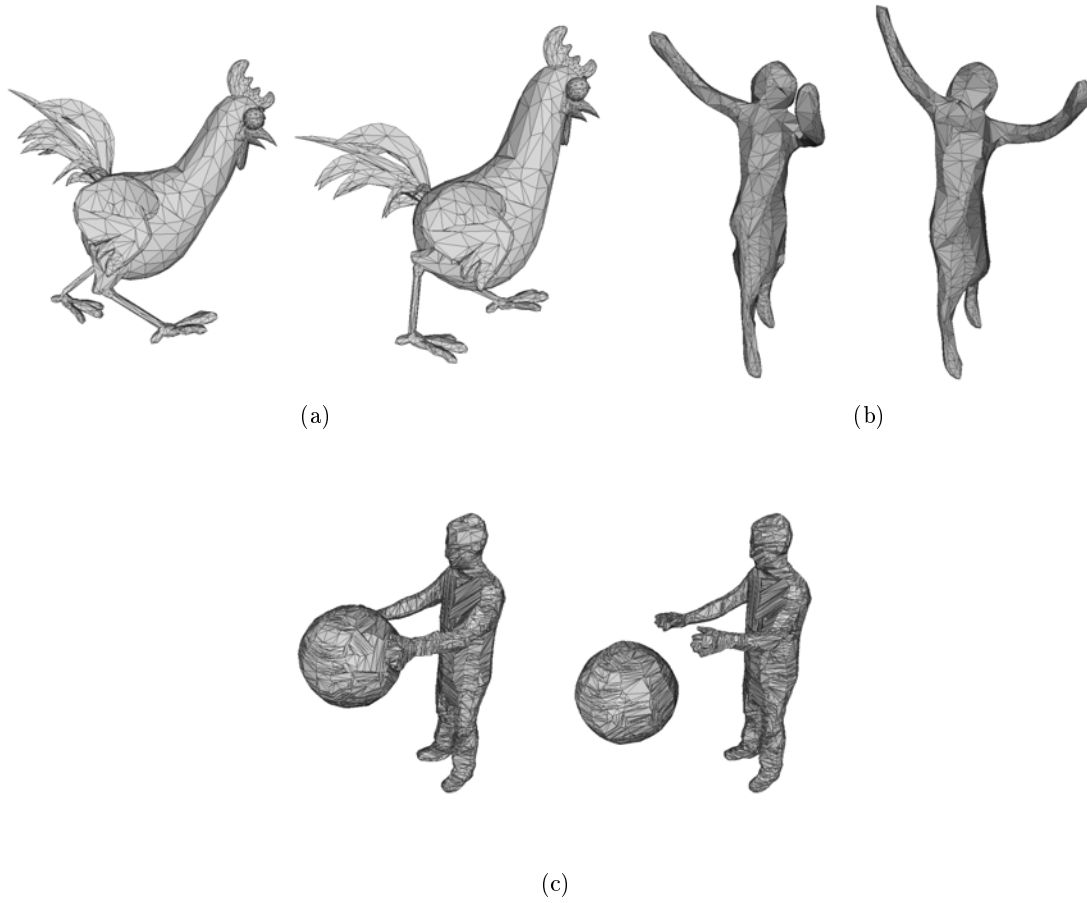


FIGURE 1.8 – **Different types of mesh sequences:** (a) Dynamic mesh (© 1996 Microsoft Corporation), (b) stable sequence, (c) unconstrained sequence (source: GRImage INRIA Rhône-Alpes & 4DView Solutions²)

time-consistent mesh where only the 3D coordinates of the vertices change through time (Figure 1.8a).

- *Stable sequence*: the number of vertices and their connectivity may change during the sequence. Therefore, the number of faces can also vary. The topology remains unchanged (Figure 1.8b).
- *Unconstrained sequence*: the number of vertices, their connectivity and the topology of the meshes may vary along the sequence (Figure 1.8c).

Model-free reconstruction methods generally produce stable or unconstrained mesh sequences. The content of the scene is reconstructed in each frame individually. In these conditions, a geometric primitive (a vertex or a triangle) in a given frame does not have any correspondence in the following frame. Topological events occur during a reconstruction due to self-intersections (collision between the limbs of the actor or with an accessory) or occlusion artifacts. As the same method can lead, depending of the actor's motions and clothing, to stable or unconstrained mesh sequences, we will only distinguish two kinds of time-varying meshes, as in [16]: **dynamic meshes** and **mesh sequences**. The use of mesh sequences in a traditional production process flow is made difficult by the volume of data and their lack of temporal coherence. The surface of the object is represented by a different sampling at each pose. This means that the connectivity of the triangular

meshes varies between two consecutive frames, which often causes flickering in lighting and shading. Finally, this lack of temporal coherence involves that a vertex cannot be tracked throughout the animation. Thus, the displacement of any actor's body part (*i.e.*, limb's trajectories) are not known. This makes the collision detection of the character and virtual object challenging and prevents the reconstructed avatar to interact with its virtual environment (physic simulation for example). Indeed, only dynamic meshes can represent the animation as a single time-evolving object, like an animated mesh made by a modeling software. Therefore, the challenge of structuring the reconstructions through time involves converting a sequence of meshes into a dynamic representation. This has a double objective: first, to provide logically organized meshes data to commercial production tools, in order to relight or redress them and second, to insert them into a controlled virtual scene.

The work described in this manuscript has been conceived for a post-production tool. The goal is to apply our algorithm on the results of the multi-view reconstruction in order to export them to post-processing tools. This way, this approach is described as an improvement of the offline reconstruction process. The online workflow is still necessary for other applications of the project, such as live onstage previsualization, but not concerned by the temporal consistency constraint. However, this project takes part in a TV production framework, which means that the whole process is sensitive to production length and that the computing time is also an important criterion for our approach. This work uses as input the result of the multi-view reconstruction, as stated in Section 1.3. Currently, the sequences of reconstructed objects are rendered thanks to a *marching cubes* algorithm [98] which extracts the surface of the volumes to obtains triangulated meshes. These meshes are textured (see Section 1.3.2). The result is therefore an unconstrained sequence of textured meshes. The stereo enhancement of the visual hull being another part of the RECOVER 3D project, lead simultaneously, our input is computed by the silhouette-based approach described previously.

1.5 Objectives

Our objective is to develop a method that yields a dynamic description of a reconstructed digital actor initially provided by a model-free, time-inconsistent process as a sequence of 3D static models. The output digital character must provide a spatio-temporal information to represent the coherent evolution of the model throughout the time of the animation. The silhouette-based reconstruction computed by the cyber-dome produces a sequence of discrete volumes which are then be transformed into a sequence of 3D textured meshes. As stated previously, these meshes are different from each other as their reconstruction is performed like a 3D snapshot of the scene, without considering the continuity with the previous or successive poses. No correspondence can thus be established between the vertices of two successive meshes. The vertex sampling, the topology and the connectivity of the reconstructed surfaces at different frames can be totally different. Using these sequences in a 3D animation and production framework is challenging because the rendering of a virtual scene containing a reconstructed character must be performed by loading the corresponding pose at each frame of the video. The digitized character cannot be tracked during the animation and therefore cannot interact with its virtual environment (collision with virtual objects for instance). Instead, we seek for an animated model, generated from these input time-series of visual hulls, which satisfies the following constraints:

1. Our output should be a unique, animated 3D model, with a time-invariant structure.

That is to say a stable triangle mesh model with constant topology and connectivity, where only the position of the vertices are moved during the animation. This representation is the equivalent of an animated virtual character, as it could be created by a 3D artist with a 3D animation software.

2. The mesh animation should be recovered from the sequences of reconstructed poses. The motion characteristics should be extracted without invasive technology and without additional capture operation. The complete process should be generic and require no user's intervention.
3. The system should be able to handle inter-frame displacements with a high amplitude. Indeed, large motions can be captured when the actor's performance includes fast gestures.
4. The mesh animation model should deal with non-rigid displacements. The advantage of multi-view reconstruction is to capture realistic motions of an actor, including his clothes, to obtain a 3D avatar as close as possible from the ground-truth displacements of a real character. Therefore, our work falls in the context of free-form animation, meaning that the displacements of our model must not be driven by a control structure. This way, the animation of the surface should not be restricted to rigid motions (rotations and translations) like in an articulated model and thus not constrained by a limited number of degrees of freedom.
5. The output mesh should not be a generic model with a predefined morphology. The digitized surface of the character can strongly differ from a human shape due to the costumes and accessories. An articulated model is thus not general enough to represent the characters we wish to reconstruct. The animated mesh should also be initialized with the shape of the captured character and be animated by a generic, free-motion method.
6. The complete process should be performed in manageable computing times. Even though it is a post-production application, the TV broadcasting framework implies limited time and resources.

1.6 Input

As described in Sections 1.3.2 and 1.5, the input sequences are obtained by a volumetric silhouette-based reconstruction. From a set of synchronized multi-viewpoint videos, a voxel grid is carved to compute the visual hull at each frame of the capture. A 3D volume is therefore reconstructed for each time step, resulting in a time series of static poses. The surface of this volumes can then be extracted as triangulated meshes, leading to mesh sequences. However, as explained in Chapters 3 and 4, the voxelized 3D objects will be used as the main geometry descriptors. In the context of the RECOVER 3D project, the initial videos are provided by the dedicated cyber-dome facilities. Nevertheless, in this manuscript, we also use other datasets from various video-based multi-view reconstruction architectures. In this case, we use the original sequences of captured images and apply the same volumetric visual hull computation to obtain volume poses, similar to the ones generated by our own system.

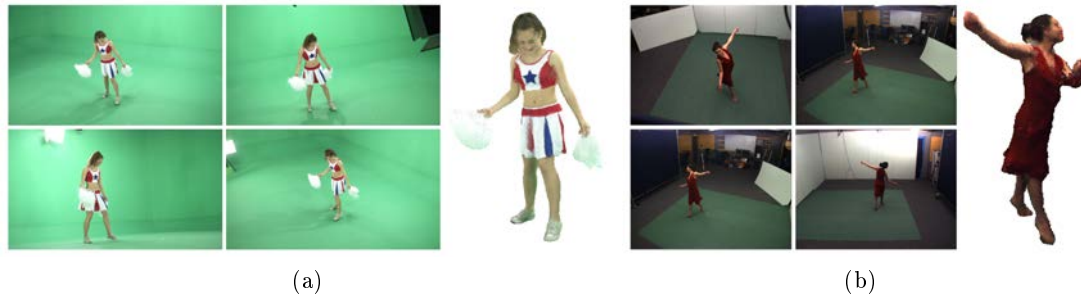


FIGURE 1.9 – **Input data.** Left to right: sample of the multi-viewpoint images and resulting volumetric visual hull after reconstruction.

1.6.1 Dataset

The input sequences used in this work were generated from various datasets of multi-viewpoint synchronized video pictures. Several actor's performances were captured with the RECOVER 3D cyber-dome. This architecture provides a set of 24-viewpoint videos. The silhouettes were extracted from each picture through a chroma-key algorithm. These silhouettes' masks were then used to compute a volumetric visual hull. After this initial model-free reconstruction, the result is a sequence of binary digital volumes, synchronized with the frames of the multi-view videos. Next, these volumes were textured using the initial video pictures. Each voxel on the surface of the reconstructed object (*i.e.*, voxels of the object directly neighboring a void voxel) is colored with RGB values computed as the average color of the pixels which correspond to the projection of this voxel in each picture from the viewpoint that see this point, at the corresponding frame (see Figure 1.9a).

Other datasets were generated using the multi-viewpoint video dataset provided by other multi-view reconstruction studios, such as *GrImage*³ [8], University of Surrey⁴ [141] and *MPI Informatik*⁵ [45] (see Chapter 2). Silhouettes masks are also provided with the original pictures. These images were used to apply our volumetric reconstruction, as described above. We thus generated the same type of volume time-series, yet with a lower precision due to the reduced amount of viewpoints available in these datasets (see Figure 1.9b). All these input data are described more in depth in Chapter 5.

1.7 Contributions

Our contribution answers to objectives, facilities and constraints of the RECOVER 3D project's industrial framework. Our method is described as an offline process to provide a time-consistent animation from an initial model-free reconstruction. This reconstruction is performed with a silhouette-based approach and already provides a real-time mode. Therefore, the main representation of the geometric shape of the actor, used as input, will be sequences of time-varying visual hull. The context of TV production and the facilities available during the studio capture also brings limitations. Our method is thus conceived

3. <http://grimage.inrialpes.fr/>

4. <http://cvssp.org/cvssp3d/>

5. <http://resources.mpi-inf.mpg.de/perfcap/>

to work on generic type of datasets. The goal of multi-view reconstruction technology is to compute a 3D representation of the actor directly from video capture, which means that the actor is filmed with its costumes and accessories. In this case our method takes no assumption about the shape to reconstruct and the type of motion. A specific template model is also inappropriate in this context. Our process also needs as little as possible user's intervention. Finally, the result must be compatible with usual post-production tools to easily integrate the reconstructed character in a virtual scene. To satisfy these constraints, our approach uses a motion estimation algorithm that does not need *a priori* knowledge on the type of motion or any type of articulated structure. In the next step, the mesh deformation allows non-rigid deformation and is not constrained by a specific morphology. The work described in this manuscript brings contributions on two major points:

1. A motion flow computation from a sequence of reconstructed volumetric objects. Our method retrieves the movement of the actor without the need of articulated structure or specifically marked points. This approach is fully automatic and non invasive, with no user intervention. It also handles non-rigid displacements and makes no assumption on the gestures. The motion vectors are directly computed from the 3D information, without the need of the original 2D pictures taken from the multi-viewpoint cameras. Our method is based on an inter-frame voxel matching to establish an initial correspondence between successive poses. The initial vector field obtained from this matching is then cleaned by a filtering operation to yield a final regularized estimation of the motion flow.
2. A time-consistent mesh animation driven by the motion flow. Our approach animates a template mesh without skeleton and without *a priori* on the shape's morphology. This method handles deformable surfaces and non-articulated displacements. It also ensures the conservation of the mesh's triangulation during the animation and follows a local rigidity prior. Our mesh deformation process is divided into several steps, starting from a global pose fitting. The last step is a local optimization to closely match the surface with the initial tracked data. This whole mesh processing method also sensitively improves the quality of the animation by providing a constant and stable triangulation and suppressing the noise and flickering from the mesh sequences used for online mode reconstruction.

The advantage of our time-consistent model is that each vertex can be tracked through the whole animation. A virtual object can then be attached to the mesh and follow it during its displacements (a virtual accessory or cloth for example). The 3D character could also be immersed in a dynamic simulation, such as particles. Besides, the connectivity remains unchanged during the motion. This enables us to work on a constant UV domain and allows, for instance, texture modification or relighting. In addition, the constant mesh structure also improves the visual quality, whereas the rendering of a mesh sequence often produces a flickering effect. It is also a mean to compress the data as the moving character is represented by a single mesh with vertices' trajectories, instead of a whole 3D object for each frame. At last, the final output of our method is exported in a standard format which is natively compatible with 3D animation software. The consistency of the mesh during the animation enables the texture mapping, allows to track the vertices through the deformation and allows to compute collision with virtual objects. A complete list of publications and communications achieved during this thesis is available in appendix A.

1.8 Overview

This manuscript is organized in 6 chapters:

- Chapter 2 provides an overview of the scientific background of multi-view reconstruction technologies. In particular, we review the different approaches which could perform the 3D reconstruction of moving shapes throughout time and using video pictures as input. We then describes more exhaustively the state-of-the-art methods for generating temporally consistent models from the reconstruction of time-evolving shapes.
- Chapter 3 presents the first step of our research, focused on the motion extraction from sequences of time-varying shapes. e describe in this chapter our 3D motion flow computation steps: a point-based inter-frame matching and a regularization process, leading to a regular motion flow.
- Chapter 4 describes our mesh animation approach. We present in this chapter our pseudo-rigid deformation method to animate a template mesh following the displacements described by the motion flow computed in the previous chapter. First, we detail an anchor's selection algorithm. Second, a mesh deformation is driven by these anchors. Third, a local optimization leads to final results.
- Chapter 5 describes the complete process flow that leads, from the original datasets, to our time-consistent animated models. We show the final results obtained with our approach. We also explore the exploitation of these results for the production of computer-generated media contents.
- Chapter 6 finally presents the conclusion of the research work described in this manuscript. We also discuss perspectives and future work.

Chapter 2

Previous work

Contents

2.1	Introduction	29
2.2	Reconstruction systems	29
2.2.1	Marker-based systems	29
2.2.2	Active sensors system	30
2.2.3	Passive sensors system	31
2.2.4	Multi-view reconstruction system	32
2.3	Classification of multi-view reconstruction methods	33
2.4	Temporally inconsistent reconstruction	34
2.4.1	Silhouette-based reconstruction	34
2.4.2	Space carving	37
2.4.3	Stereo-based reconstruction	37
2.4.4	Other approaches	39
2.4.5	Application to dynamic reconstruction and limitations	39
2.5	Temporally consistent reconstruction	39
2.5.1	Dynamic shape registration	40
2.5.2	Non-realistic model-based methods and motion tracking	41
2.5.3	Realistic model-based methods	44
2.5.4	Temporally consistent model-free methods	46
2.6	Conclusion	48
2.6.1	Discussion	48
2.6.2	Our approach	49

Résumé

Ce chapitre présente un état de l’art des différentes méthodes de reconstruction multi-vues qui permettent de reproduire sous forme d’un modèle 3D la performance d’un acteur filmé par un ensemble synchronisé de caméras positionnées à différents points de vue. Notre étude cible les méthodes non invasives (sans marqueurs tels que ceux utilisés par les technologies de *capture de mouvements*) et utilisant des capteurs passifs (caméras vidéo uniquement). Nous distinguons les résultats de ces différentes techniques de reconstructions selon un critère de *cohérence temporelle*, à savoir :

- Les reconstructions *temporellement incohérentes* produisant une succession de modèles 3D au cours du temps sans qu’aucune correspondance ne soit établie entre leur géométrie.
- Les reconstructions *temporellement cohérentes* générant au contraire des modèle 3D animés et dont la structure reste stable au cours du temps. Il s’agit le plus souvent d’un maillage dont la position des sommets évolue au cours du temps tout en conservant la connectivité.

De plus, les méthodes de reconstruction multi-vues sont habituellement réparties en deux catégories :

- Les *méthodes libres* se distinguent par le fait qu’aucune connaissance *a priori* (nombre et nature des objets, morphologie du ou des personnages) n’est fournie au système. Du fait de leur généricité, ces techniques génèrent des résultats dépourvus de cohérence temporelle : en effet, une reconstruction statique est calculée pour chaque *frame* indépendamment des autres.
- les *méthodes basées modèle* disposent d’une géométrie de référence (maillage triangulaire par exemple) de l’objet à reconstruire. La reconstruction consiste alors à faire évoluer ce modèle préétabli en fonction des données issues de la capture multi-vues (suivi de silhouettes ou de flots optiques par exemple). Ces méthodes sont plus robustes que les méthodes libres et ont l’avantage de générer des données avec une forte cohérence temporelle. Néanmoins, elles sont restreintes par la morphologie du modèle.

Nos données d’entrées sont des séquences d’*enveloppes visuelles* issues de reconstructions basées silhouettes (méthode libre) et dépourvues de cohérence temporelle. Notre objectif est d’en extraire un maillage animé unique et temporellement cohérent. La plupart des techniques permettant de générer de tels résultats sont des approches basées modèle qui utilisent le plus souvent des géométries articulées [75, 17, 43] limitées au suivi de mouvements rigides. Des modèles hybrides mélangent animation squelette et surfaces déformables pour reconstruire également les mouvements libres tels que ceux des vêtements par exemple. Afin de produire des résultats plus réalistes, d’autres méthodes utilisent un modèle spécifique à l’acteur filmé [45, 164, 61]. Dans ce cas, une étape préliminaire est nécessaire afin de générer un modèle adapté à la morphologies et/ou au costume de l’acteur (*via* un scanner 3D par exemple). Pour contourner les limitations des modèles articulés, certains modèles utilisent des déformations par *cage* [56] ou des ensembles de *patches* [33]. Enfin, des méthodes sans modèle utilisent des algorithmes de recalages non-rigides [145, 126, 25]. Un ensemble de correspondances permet alors de guider la déformation d’une surface de manière à suivre une succession de poses issues de reconstructions temporellement incohérentes. La plupart de ces méthodes sont cependant sensibles aux mouvements de grande amplitude.



FIGURE 2.1 – Marker-based motion capture (source: 3IS - Institut International de l’Image et du Son)

2.1 Introduction

This chapter presents the technical and scientific background of research on multi-view reconstruction. It first describes the usual kind of motion capture and 3D reconstruction technologies of animated objects in Section 2.2. Then, a review on the video-based methods which reconstruct time-varying surfaces, using a set of multi-viewpoint video cameras, is presented in Sections 2.3 and 2.4. Finally, the various state-of-the-art approaches which produce time-consistent geometries of animated objects from unconstrained multi-view captures are discussed in Section 2.5.

2.2 Reconstruction systems

We first provide an overview of the various types of 3D capture systems, starting with usual *marker-based* motion-capture in Section 2.2.1. We then describe in Sections 2.2.2 and 2.2.3 several reconstruction systems which use *active* or *passive* sensors. We finally focus on video-based multi-view reconstruction with markerless, passive sensors systems in Section 2.2.4. The RECOVER 3D system belongs to this last category.

2.2.1 Marker-based systems

Traditional motion capture relies on physical markers, positioned at a set of key points on the actor wearing a neutral suit (see Figure 2.1). These markers are tracked in the 3D space by a set of specific cameras. Most of the time, the use of infra-red cameras and infra-red reflecting markers ensure that the capture will not be interfered by exposition variations. The trajectories of the markers are then be transfered to a specific mesh model. This mesh is therefore animated, following the captured motions of the actor. *Skeleton-based* animation is one of the most widely used mesh animation method for human morphologies [79, 89] and is particularly suited for marker-based motion capture. The mesh is *skinned*

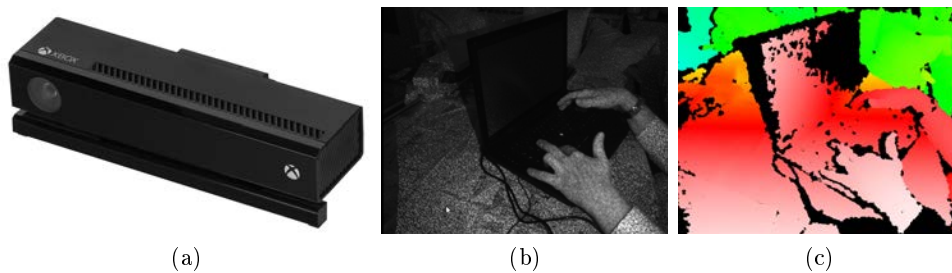


FIGURE 2.2 – **Example of active sensor system: Kinect camera** (a) kinect sensor, (b) structured light infra-red pattern, (c) resulting depth-map (source: wikimedia).

to the skeleton, which means that the vertices are associated to the corresponding section of the skeleton and move according to this *bone*. The displacement of each vertex is thus interpolated from the skeleton's motion which is, in the case of MoCap, driven by the markers displacements.

The same principle has been adapted for reconstruction of the face expression. Markers are positionned on the actor's face (usually painted dots) and captured by a camera mounted on helmet. This technology, referred to as *perfcap* (PERformance CAPture), enables to animate a facial model with predefined specific points which produce complex animation of a surface [105], less constrained than a skeleton articulation. Park and Hodgins [119] applied the same kind of method to a whole body capture. A set of more than 300 reflective markers are placed on the actor. The tracking of these points, associated with a specific parametric human model, captures the motion of the skin itself (*i.e.*, the actor's *surface*) instead of just an articulated skeleton's pose.

All these systems use *passive markers* that only reflect the information from the light source. Other systems use *active markers*, made with LEDs, which emit their own light to highlight their position in space.

2.2.2 Active sensors system

Other systems capture the actor's performance without marker constraints, using active sensors. An active sensor sends itself a specific signal and captures the returned information. Active cameras produce a specific type of light and interpret the returned image accordingly. The first family of approaches is based on *structured light* [59]. Using a projector coupled with a camera, a predefined pattern image (lines or grid for example) is projected on the object (*i.e.*, the actor) of which the camera takes a picture. By analyzing the deformation of the pattern in this picture, a 3D shape can be computed. A complete reconstruction of the scene can be performed by repeating this operation from several view-points. With this method, the reconstructed surface cannot be textured by a color capture of the real scene. To avoid the pollution of the scene by this light in the visible spectrum, other systems use an infra-red projection, coupled with a specific infra-red sensor. As an example, we can cite the well-known Microsoft Kinect⁶ [175]. This device combines an infra-red projector which produces a *speckle* pattern, an infra-red camera and a usual color camera, as presented in Figure 2.2. The result of the infra-red reconstruction is a

6. <https://dev.windows.com/en-us/kinect>

depth-map of the scene. The second type of active sensors are the *Time-of-Flight* cameras which produce a light signal, with a known speed, and compute the difference between the emission and the captured reflection. It can be compared to LIDAR systems, except that the whole scene is captured by the light pulse. During a LIDAR scanning, a set of points are captured by sending a laser in the scene and computing the time of flight in the same way, leading to a point cloud reconstruction. The advantage of these systems is that they natively produce a 3D reconstruction of the scene. However, the information is produced in the form of point clouds or depth-maps, therefore the recovering of the surface of the object may be subject to an interpretation process. The main disadvantage in our case is that, in a multi-viewpoint system, each sensor produces a reconstruction as a depth information from each viewpoint. The reconstruction of the whole scene then requires a complex merging operation of these data, particularly as they may be noisy. At least, all these active sensors need a strongly controlled environment to avoid light pollution, and are sensitive to specular elements.

Recently, new 3D scanner devices (*e.g.*, Kinect) allow a fast reconstruction of dynamic scenes, with a high frame rate but a limited spatial resolution [130, 172, 69]. This leads to an important overlap in the adjacent frames of the resulting point-cloud time series. Other systems with higher resolution produce more accurate data, but with significantly larger inter-frame deformation. All these scanning technologies produce partial point-clouds, since each sensor can only provide a limited field of view, like shape-from-stereo reconstruction (see Section 2.4). To compute a complete surface, several acquisitions, synchronized from different viewpoints, are performed simultaneously. The point-clouds reconstructed from each capture unit can then be aligned (*e.g.*, using an ICP algorithm [21, 37]) and merged to produce a single object at each pose. The ICP algorithm minimizes the difference between two point clouds by iteratively applying rigid transformations. These devices are often used to reconstruct a dynamic scene and are the main alternative to video-based multi-view approaches (see Section 2.4). An extensive number of state-of-the-art methods that compute a dynamic reconstruction are based on these scanning technologies. Even if they are not directly linked with our framework, these point-based approaches can often be adapted and/or compared to other types of input data. As these reconstructions often suffer from holes in the acquisition, the filling of these inconsistencies is also an important step to obtain a watertight surface (see Section 2.5). In the case of our framework, the optical quality and image resolution of the usual depth-cameras are still lower than video cameras, which is harmful for the quality of textures. In addition, the depth acquisition does not exceed 10m and the accuracy noticeably decreases beyond 5m, which is not sufficient for a 20-meter large studio as the one we use.

2.2.3 Passive sensors system

The multi-view reconstruction systems used in this project belong to the family of passive sensors systems. In the case of video information, a passive sensor is a simple video camera. The video reconstruction is made possible by using several viewpoints to shoot the scene, as described in Section 1.2. The reconstruction can be achieved by several kinds of methods, using photometric informations. The *Virtualized Reality* system developed by Kanade *et al.* [73] is one of the oldest example (see Figure 2.3a). The most common and generic ones use the silhouette pictures or stereo-matching to recover the shape of the actor, as described in Section 2.4.

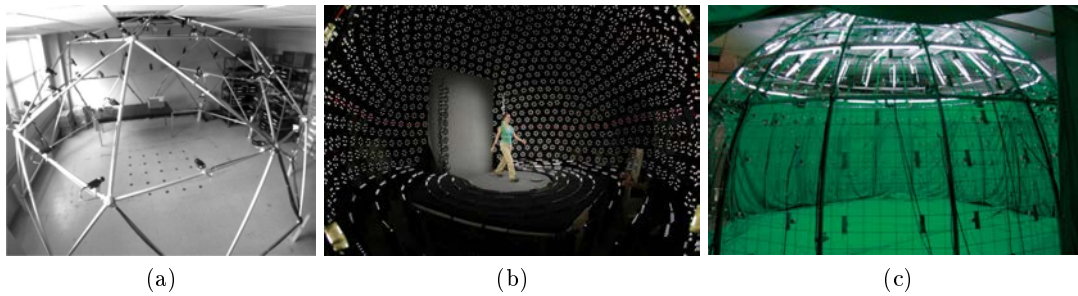


FIGURE 2.3 – **Video-based reconstruction systems (passive sensors):** (a) Virtualized Reality, (b) Light Stage, (c) Tsinghua University

Others systems perform the reconstruction by recording the same scene under several illumination conditions. For example, the *Light Stage*⁷ system [165] contains height high frequency cameras and a set of 1200 light sources (see Figure 2.3b). By acquiring the actor under several illuminations in a highly reduced time, this methods computes a set of normal maps by a *shape-from-shading* (or *photometric stereo*) algorithm, that are merged into a single 3D shape by a multi-view matching. This system has been adapted in several versions, allowing a complete reconstruction of the actor or a close capture of the face. It has been used in the production of several movies. Another example is the Multi-camera multi-lighting dome of Tsinghua University composed of 30 cameras (including 10 high-speed cameras) and 310 LEDS divided in 31 directional lightings (see Figure 2.3c). This structure is used for stereo-based point cloud reconstruction [96]. Other types of systems are dedicated to marker-less facial motion capture by tracking a set of features, like lips and eyes for example. The *Digital Ira* project (Activision and USC ICT [5]) is dedicated to facial expression capture and uses the Light Stage X system. At last, *Panoptic Studio*⁸ at Carnegie Mellon University is composed of 480 cameras, distributed on 20 panels of 24 cameras each.

2.2.4 Multi-view reconstruction system

The infrastructure of the reconstruction systems described in this section can be compared to the RECOVER 3D cyber-dome. These systems are only based on a set of common video cameras. The reconstruction of the objects' surface only uses the colorimetric information (*e.g.*, silhouettes, stereo-matching or optical flow) without deducing informations from other objects like light sources (as in shape-from-shading) or markers. Several projects use such kind of virtual studios.

The University of Surrey *4DVT* (4D Video Textures) project^{9 10} [141] uses a set of 10 cameras, placed on a ring of 8-meter diameter and 2-meter height, with a blue-screen backdrop. That provides a $4 \times 4 \times 2$ -meter volume capture (see Figure 2.4a). The *MR-PreViz*¹¹ system (Kyoto University) [149] uses 12 cameras distributed on two rings of 6m diameter at heights of 1.2m and 2.2m, respectively, providing a $3 \times 3 \times 2$ -meter capture

7. <http://gl.ict.usc.edu/LightStages/>

8. <http://www.cs.cmu.edu/~hanbyulj/14/visibility.html>

9. <http://www.cvssp.org/projects/4d/4DVT/>

10. <http://cvssp.org/projects/4d/HybridSurfaceMotionGraphs/>

11. <http://vision.kuee.kyoto-u.ac.jp/MR-PreViz/index.html>

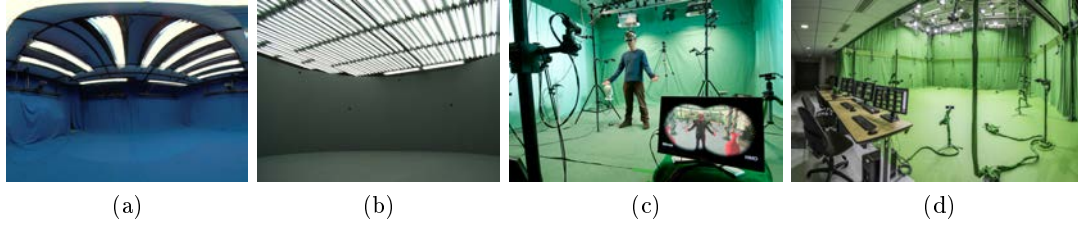


FIGURE 2.4 – **Multi-view reconstruction systems:** (a) University of Surrey (b) MR-PreViz, (c) GrImage, (d) Kinovis

volume. Two other cameras are placed above the stage, at the ceiling. A last one is dedicated to capture the face of the actor. We can also cite the *GrImage* platform^{12 13} (Grenoble University) [8]. The recent project *Kinovis*¹⁴ is an evolution of this last system, using a large set of 68 cameras in a $105m^3$ chroma-key studio (see Figure 2.4d).

The common elements of multi-view reconstruction studios are a set of video cameras, which positions around the capture area maximize the viewpoints. The silhouette’s extraction is computed with a chroma-keying process or another method of *background subtraction* (or *foreground detection*, *i.e.*, difference of the capture image with a background model). Other silhouette extraction methods can also be employed, like *contour-based methods* (using edges detection filters, *e.g.*, gradient thresholding) or *region based methods* (*e.g.*, histogram segmentation or region growing). Most of these systems use a silhouette-based or a stereo-based reconstruction as the main prior for more advanced motion tracking operations.

2.3 Classification of multi-view reconstruction methods

In the following sections, we distinguish the multi-view reconstruction approaches according to their temporal consistency:

- **temporally-consistent reconstructions** result in dynamic models which structure stays unchanged during the animation. It is usually a dynamic mesh (see Section 1.4) which connectivity remains constant.
- **temporally-inconsistent reconstructions** produce several 3D objects throughout the time interval of the capture with no continuity in their structure. The result is most of the time a mesh sequence (see Section 1.4) or time-series of point clouds.

These two type of reconstructions are detailed in Section 2.4 and 2.5, respectively. The initial reconstruction performed by the RECOVER 3D studio, described in Section 1.3.2, produces temporally inconsistent results. Our goal is to transform these mesh sequences into temporally consistent mesh animations. The temporal consistency is also strongly linked to the two main families in which multi-view reconstruction approaches are usually split into:

- **Model-free methods** can be distinguished by the fact that no prior knowledge (relating to the nature and number of objects, characters’ morphologies) is provided to the system. The most common techniques (shape-from-silhouettes and shape-from-

12. <http://grimage.inrialpes.fr/>

13. <http://www.4dviews.com/>

14. <http://kinovis.inrialpes.fr/>

stereo) belongs to this family. Due to their general nature, these techniques generate results without any temporal coherence. Indeed, a reconstruction is calculated for each frame independently of each other (see Section 2.4). These methods are also efficient in term of computing time. A real-time computation can even be reached for low resolution reconstructions. More recent model-free methods have been proposed to perform temporally consistent reconstructions, even if they loose the advantage of efficient computing due to their higher complexity (see Section 2.5).

- **Model-based methods** use a reference geometric model (for example a triangle mesh) of the object to be reconstructed. This prior knowledge can be manually constructed or obtained using an acquisition system (for example a 3D scanner). Reconstruction involves evolving the reference model in relation to data taken from multi-view capture (silhouettes, optical waves, etc.). These methods are more reliable than model-free methods and exhibit the decisive advantage of generating data with strong temporal coherence. However, due to the use of a reference model, they are, in the majority of cases, restricted to reconstructing a single subject with human morphology.

2.4 Temporally inconsistent reconstruction

The most common model-free approaches produce temporally inconsistent reconstructions. These methods are based on a static reconstruction, using pictures of an objects from multiple viewpoints. Applied to performance capture of dynamic scenes, these methods only repeat the same reconstruction process at each frame of the video sequences, using a set of synchronized cameras. This reconstruction is performed at each frame, thus reconstructing the pose of the actor at this time. The result is a sequence of static 3D objects (usually textured meshes) which represent the successive poses of the actor. This type of dataset is sometimes referred to as *3D video*, as an analogy with the successive static frames of a common video. These sequences can then be exported to a virtual scene, like common 3D objects. The animation can be reproduced by successively loading and rendering the poses, following the frame rate of the original videos. However, as described in Section 1.4, these time series do not have any temporal continuity. Since each pose is reconstructed independently, the successive objects have different structures (*e.g.*, different numbers of vertices and different connectivity). The topology of the shape may also vary from one pose to another. Following the formalism given in Section 1.4, the result of this kind of reconstruction is most of the time a stable or unconstrained mesh sequence. The following sections describe these methods. The algorithms described in Sections 2.4.1 and 2.4.2 compute the maximum 3D shape which correspond to the captured images. Sections 2.4.3 and 2.4.4 present other methods using correspondences between the pixels in the images taken from various viewpoints to recover the 3D position of the associated points.

2.4.1 Silhouette-based reconstruction

As stated in Chapter 1.2.2 (see Section 1.2.3), visual hull reconstruction methods often belong to one of these two main families: volumetric approaches and surface-based approaches. Volumetric approaches are based on a discretization of the scene’s space into a 3D grid, usually regular, where each cell is named a *voxel* (VOLUME ELEMENT). These voxels

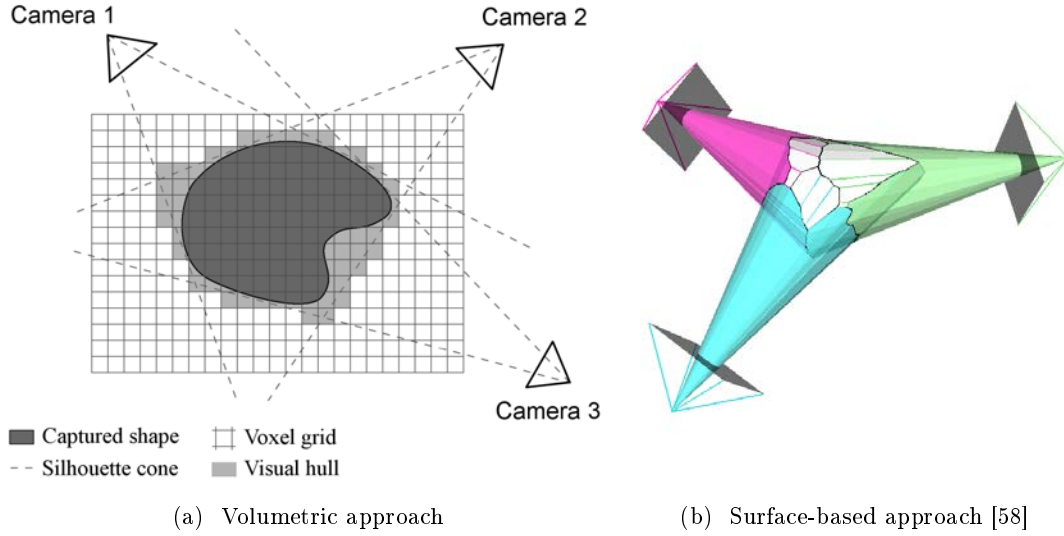


FIGURE 2.5 – Visual hull reconstruction

are binary labelled as *internal* or *external* according to their position, inside or outside the figured object. The group of internal voxels thus describes the volume of the object. In a volumetric visual hull computation, this volume is iteratively carved according to the silhouette masks extracted in the multi-view cameras [106, 40]. All the voxels are projected toward a camera and compared to the associated silhouette picture. If a voxel is projected on a pixel inside the silhouette, it is labelled as internal, otherwise, it is labelled as external. The volume of the scene is thus *carved* according to the silhouette cone of this camera. By repeating this operation with all the cameras and maintaining a voxel as external when it has been labelled like this for at least one silhouette, the volume contained in the visual hull is computed (see Figure 2.5a and 2.6a). Several improvements of this approach lead to efficient computation of a voxel-based representation (*e.g.*, [146, 116, 39, 137]). The surface of this volume can then be extracted, using for example a *marching cubes* algorithm [98], to obtain a 3D mesh. The surface-based (or polyhedral) approach directly constructs a 3D mesh by computing the intersection of the silhouette cone's surfaces [84]. Several proposed techniques compute local surface patches [143] or strips [108]. More recent methods directly compute a triangulation from the captured silhouettes' edges [58]. For each pixel of a silhouette's edge, the line that links this pixel with the optical center of the corresponding camera contains all the potential positions of the corresponding point in the 3D scene. The union of all these lines defines the silhouette cone of a silhouette. The intersection of these silhouette cones is recovered by computing the intersection points between the lines of different silhouettes. A triangulation of this shape is computed afterwards (see Figures 2.5b and 2.6b).

The main strength of silhouette-based reconstruction is its simplicity. The basic methods are easily implemented and accelerated, so that real-time reconstruction can be reached. Although it only approximates the shape, the estimation it provides is suitable for a number of purposes. Once textured, the visual hull offers a convincing rendering, especially for a static object. The main disadvantage of shape-from-silhouette is that it cannot reconstruct some details on the object's surface. This type of reconstruction is unable to recover the concave details of the surface which cannot be observed in a silhouette, re-

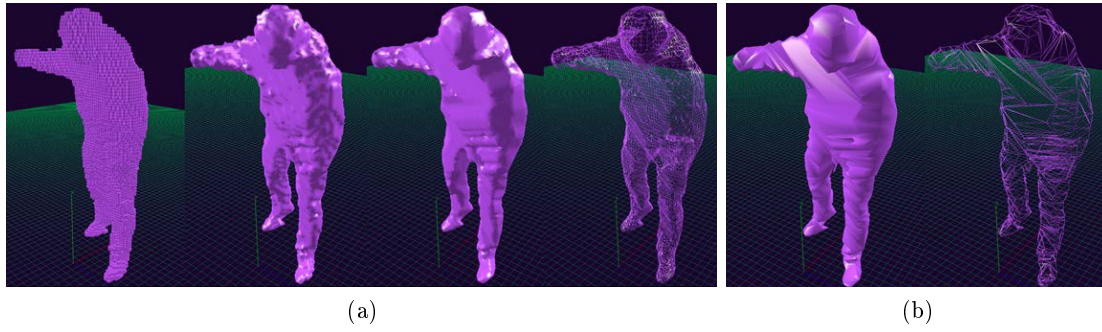


FIGURE 2.6 – **Visual hull reconstruction:** (a) Volumetric approach. From left to right: raw voxel grid, *marching cubes* mesh, Adaptive Marching Cubes (solid and wireframe rendering), (b) Surface-based approach (solid and wireframe rendering) – (source: XD Productions).

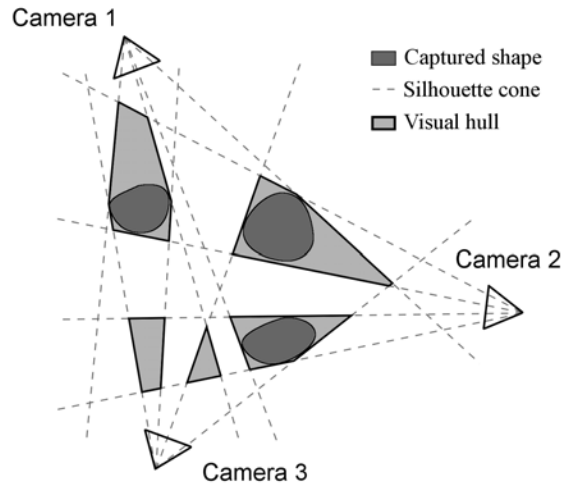


FIGURE 2.7 – Visual hull limitations. Note the phantom volumes produced by occultations.

regardless of the viewpoint on the object. The quality of the reconstruction also depends on the number of viewpoints. Several details on the surface of the object can be missed if they do not appear on one of the silhouette. Similarly, self-occlusion (especially due to the limbs in the case of actor's reconstruction) can lead to topology inconsistencies. If several objects are present in the reconstructed scene, they may not be properly identified in the final shape. These limitations are illustrated in Figure 2.7. Another important limitation is the lack of robustness of the visual hull computation to noisy input. Every erroneous data appearing in at least one silhouette mask can lead to an artifact in the final surface. The quality of the silhouettes' contour is therefore a critical element. Several techniques have been developed to increase the robustness of silhouette-based reconstruction, using for instance *graph-cut* optimization [29]. However, these more elaborate methods lose the simplicity and real-time capability which make these approaches attractive.

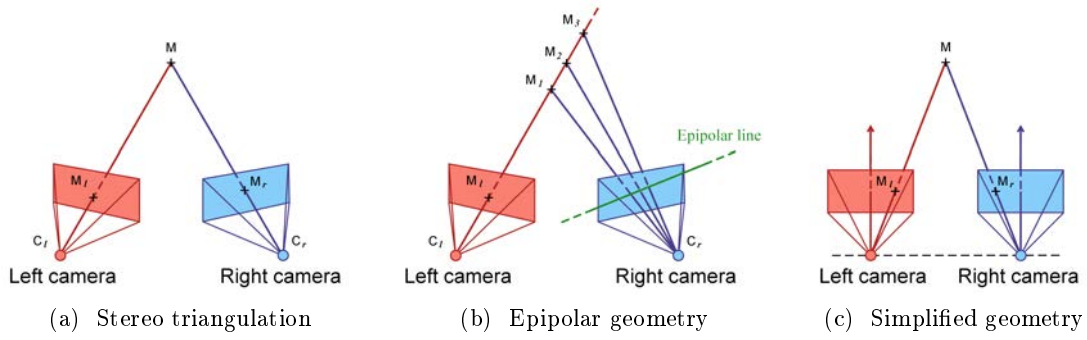


FIGURE 2.8 – Binocular geometry for stereo-based reconstruction.

2.4.2 Space carving

In order to fix the existing limitations of the shape-from-silhouette approaches, another set of methods uses color information from each view to select the voxels within the bounding volume (based on volumetric approaches). The *voxel coloring* technique, proposed by Seitz and Dyer [136], involves subdividing the regular grid of voxels into successive layers, from the nearest to the farthest in relation to the cameras (the cameras being set out in a semicircle around the object to reconstruct). Voxel coloring is based on the hypothesis that a voxel on the surface of an object must have the same color in each view, known as a *photo-consistent* voxel. The *space carving* principle, introduced by Kutulakos *et al.* [81], can be seen as an extension of voxel coloring adapted to an arbitrary camera setup. This relies on sweep planes aligned with the three principal axes X, Y and Z. Only the cameras behind the sweep plane are used to manage the occlusion. According to Kutulakos, a voxel is not visible by a camera if it is out of the view frustum or if it is occluded.

2.4.3 Stereo-based reconstruction

Another important family of model-free reconstruction is the *shape-from-stereo*. These methods are based on a pixel matching between images from different viewpoints. These pixels can then be replaced in the 3D space. As mentioned in Chapter 1, each pixel from one image must be matched to a pixel in another image, according to a given correlation criterion, *e.g.*, *Sum of Absolute Differences* (SAD) or *Sum of Squared Differences* (SSD). See [133] for a taxonomy of usual correspondence algorithms. Knowing the position of the optical center of a camera and the projection of a point (*i.e.*, pixel) on this camera, the ray emitted from the optical center through the pixel defines an *optical line*. By repeating this operation on two cameras, the intersection between the two resulting lines gives the position of the 3D point using, for instance *Mid-point* or *Direct Linear Transformation* (DLT) methods [1]. This operation is called *triangulation*. As depicted in Figure 2.8a, given two video cameras with optical centers C_l and C_r , the point M from the captured scene is projected toward M_l and M_r . The position of M is recovered by computing the intersection of the lines $(C_l M_l)$ and $(C_r M_r)$. To simplify the matching process, the *epipolar geometry* allows to reduce the dimension of the correspondence searching for each pixel by finding its corresponding *epipolar line*, which contains the potential position of the homologue pixel, in other images. See [63] for details about multiple view geometries. As shown in Figure 2.8b, the corresponding point of M_l in the right image lies on the epipolar

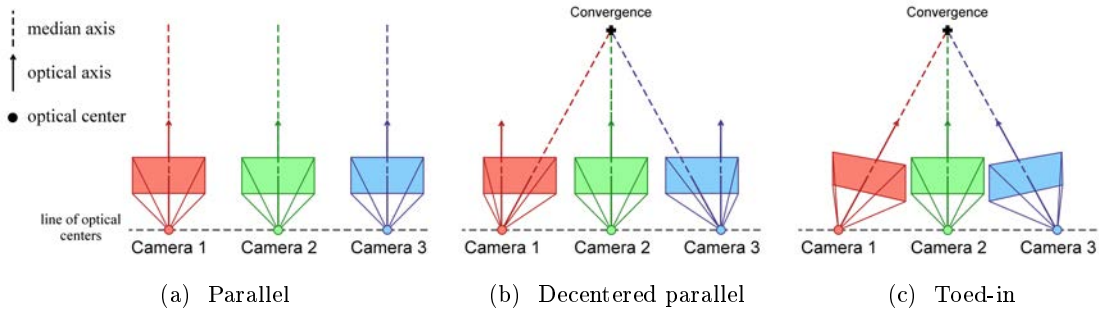


FIGURE 2.9 – Different types of multi-baseline geometries for multi-stereo capture.

line, i.e., the projection of $(C_l M_l)$ in the right image.

Thus, the result of the reconstruction can be represented as a point cloud or a depth-map. In a multi-view reconstruction context, where multiple cameras are disposed around the captured scene, several stereo reconstructions can be performed using pairs of cameras, leading to a set of point clouds. These point clouds represent parts of the object, depending on the acquisition viewpoint. These partial reconstructions can be merged afterwards to define a single 3D shape. The stereo capture can also be performed in a *simplified geometry* where image planes are parallel one to another with the same focal length and vertical pitch (see Figure 2.8c). The matching pixels then have the same row ranks in both images. This geometry is named *binocular* if it contains only two cameras. However, in the case of multiocular geometries, we can extend this approach to a higher number of cameras (more than two), leading to a *multi-baseline* geometry, as presented in Figure 2.9. These multi-baseline geometries can be classified in three types of layouts with aligned optical centers:

- *Parallel*: the frustum is horizontally centered on the associated optical axis (see Figure 2.9a).
- *Decentered parallel*: the frustum is not anymore centered on their optical axis. The median axis converge at the same 3D point (see Figure 2.9b).
- *Toed-in*: the optical axis of the cameras converge at the same point in the 3D space (see Figure 2.9c).

In the RECOVER 3D project, a complete reconstruction can also be obtained by using several multiscopic units of cameras at different positions around the object (see Chapter 1). These blocks are based on multi-baseline geometries and produce several point clouds or depth-maps which then have to be merged to define a single 3D shape.

The multi-view stereo reconstruction is a widely used approach to perform a reconstruction from multiple viewpoint videos [135]. The shape-from-stereo is also very sensitive to erroneous data. The quality of the reconstruction is directly linked to the quality of input pictures. Depending on the chosen matching approach, wrong matching can be produced, due to exposition difference between the pictures for example. These errors lead to noisy point clouds as output. The stereo-matching quality also depends on the redundancy of the features in the pictures. It is therefore sensitive to poorly textured surfaces (plane surfaces with uniform color). As the shape-from-silhouette provides a convex hull which contains the objects, it can be seen as a good initialization for more precise reconstruction algorithms such as shape-from-stereo. Therefore, many approaches are based on a mix between these two technologies (*e.g.*, [38, 57, 140]). The outliers of the point cloud

obtained by stereo-based reconstruction can be removed if they are not contained in the visual hull. A visual hull can also be directly carved by stereo reconstruction. This type of hybrid reconstruction is considered in the RECOVER 3D project to produce a high quality model-free reconstruction.

2.4.4 Other approaches

Other 3D reconstruction approaches, such as *Structure-from-motion* or *Photogrammetry* algorithms also rely on multi-viewpoint geometry acquisitions and use triangulation operations to recover 3D points from a set of images. However, in these approaches, the camera geometry is not predefined like in most of the methods previously described. Structure-from-motion approaches usually extract 3D features from multiple images, acquired from multiple viewpoints using, for instance, SIFT (*Scale-Invariant Feature Transform* [99]), DOG (*Difference-Of-Gaussians* [100]) or SURF (*Speeded Up Robust Features* [20]) algorithms. A correlation operation is applied to match these features from one image to another. The result is a sparse set of correspondences which are used as landmarks to recover an epipolar geometry. A dense reconstruction is then computed through usual triangulation techniques.

2.4.5 Application to dynamic reconstruction and limitations

Although these model-free systems have been first used for the reconstruction of static scenes, using pictures taken from different viewpoints, they have been applied to the reconstruction of dynamic scenes, in particular for actor's performance capture. However, the reconstruction process remains unchanged. The multi-view cameras are synchronized in their framerate and a new reconstruction is performed at each frame from the corresponding pictures at each viewpoint. The final result is a static reconstruction (mesh, point cloud or volume) for each frame. The animation is represented by successively rendering the poses of this sequence of 3D object (or 3D video). These sequences are most of the time transformed to 3D meshes for a convenient rendering. These mesh sequences are also named *time-varying meshes* because of the variations in the meshes' structure appearing from one frame to another, since the actor is reconstructed at each frame without continuity with the other poses.

2.5 Temporally consistent reconstruction

This section describes the reconstruction approaches that provide a single and stable object throughout the time of the synchronized multi-viewpoint videos. The result of such reconstruction contains a temporal information that allows to track an object or a feature's position during the 3D animation. Many of these approaches focus the tracking of human motion, for automatic action recognition or virtual reality for example. More advanced techniques are merged with 3D acquisition to obtain a robust and realistic reconstruction of a dynamic scene. We also discuss the viability of these techniques in the context of the core contributions of our approach as stated in the introduction.

		Input	
	animation prior	3D scan (point-cloud sequences)	3D video-based reconstruction (temporally inconsistent sequences)
Model-based	Articulated		multi-viewpoint videos and/or depth-maps
Model-based	Articulated		Kehl & Gool [75] Horaud <i>et al.</i> [64] Corazza <i>et al.</i> [42] Corazza <i>et al.</i> [43] Luo <i>et al.</i> [102] Vlastic <i>et al.</i> [164] Gall <i>et al.</i> [61] Liu <i>et al.</i> [97]
			Plankers & Fua [125] Carranza <i>et al.</i> [34] Balan <i>et al.</i> [17] De Aguiar <i>et al.</i> [46] De Aguiar <i>et al.</i> [47] Theobalt <i>et al.</i> [156] Ahmed <i>et al.</i> [2]
Model-based	Free-form	Anguelov <i>et al.</i> [12] Li <i>et al.</i> [90]	Kilner <i>et al.</i> [76] Duveau <i>et al.</i> [56] Cagniard <i>et al.</i> [33] Allain <i>et al.</i> [7]
			de Aguiar <i>et al.</i> [45]
Model-free	Articulated	Zheng <i>et al.</i> [176] Mukasa <i>et al.</i> [112] Pekelný & Gotsman [122] Chang & Zwicker [36]	
Model-free	Free-form	Mitra <i>et al.</i> [110] Süßmuth <i>et al.</i> [145] Wand <i>et al.</i> [168] Wand <i>et al.</i> [167] Popa <i>et al.</i> [126] Tevs <i>et al.</i> [153] Li <i>et al.</i> [91] Bonarrigo <i>et al.</i> [25]	Starck & Hilton [140]

TABLE 2.1 – **Taxonomy:** overview of temporally consistent reconstruction approaches cited in this document (restricted to full-body capture and markerless methods).

Table 2.1 lists the methods described in Sections 2.5.1 to 2.5.4. Several kinds of *Markerless Motion Capture* (MMC) have been developed recently, encouraged by the easy availability of Microsoft Kinect sensors. This technology provides, among other things, an automatic human skeleton fitting on the videos and depth-maps natively captured by the hardware. Table 2.1 shows that many methods use this type of 3D scan time series as input. The second category works on sequences of video-based multi-view reconstructions (*e.g.*, visual hulls). The third family of approaches uses directly a set of multi-viewpoint videos and/or depth-maps as input.

2.5.1 Dynamic shape registration

Computing a correspondence in a time series of 3D objects can be seen as a shape registration problem [161, 35]. In our case, since the input data is a sequence of a reconstructed object acquired at different time steps, this is often called a *dynamic registration*

or *time-varying registration*.

2.5.1.1 Rigid registration

The most usual registration methods, such as ICP, only rely on a rigid registration [21, 37]. This algorithm is often used, for instance, to align and merge several point-clouds that represent the same object acquired from different viewpoints, like stereo-based scanning in multi-view reconstruction. However, the dynamic shapes like the actors in our case are not limited by a single rigid transformation (*i.e.*, a static object which is only translated and/or rotated) but deform their surface between the successive frames of the reconstructed sequence. Therefore, the reconstruction of time-varying subjects assumes a *non-rigid registration* (or *deformable body registration*).

2.5.1.2 Non-rigid registration

Several methods rely on non-rigid variants of rigid alignment algorithms (*non-rigid ICP*) [62], by computing an explicit set of correspondences between the two shapes [110, 168] (point to point for example). For the registration of several poses over a complete sequence of time-varying shapes, some approaches use a template shape which is successively matched with each shape [90]. Other template-free methods directly compute the correspondences between adjacent frames [154, 126, 167]. For the sequences that contain large deformations between consecutive poses (*e.g.*, due to a fast motion or a low framerate), the limited overlap of the adjacent shapes leads to make assumptions on the deformations, as an *a priori* knowledge. For example, the captured subject can be supposed articulated. The registration thus relies on rigidly moving clusters [10, 13]. More complex prior assumption on the shape's motion can be represented by a complete template model specific to the captured subject and given as input to the method (*e.g.*, a human model for actors' reconstruction) [23, 121, 11]. More generally, the dynamic registration can be modeled like a global energy minimization. This energy takes into account both a data matching term (or fitting error) and a prior term (*e.g.*, smoothness or local rigidity) [28]. Using this approach allows to compute free-form registration without prior assumption on the time-varying surface [92, 68].

2.5.2 Non-realistic model-based methods and motion tracking

The pose recognition of human bodies from videos is a widely studied domain (see, for instance, [127, 171]). These methods can be extended to 3D motion tracking in a multi-viewpoint context. The technologies described in this section can be related with marker-less motion capture given that their goal is not to compute a realistic reconstruction of an actor but recovering the pose and motion of a human subject.

2.5.2.1 Skeleton-based pose-tracking

Many methods rely on a generic human body parametric model, taken as an input. Mukasa *et al.* [112] described a method to recover a kinematic structure from a sequence

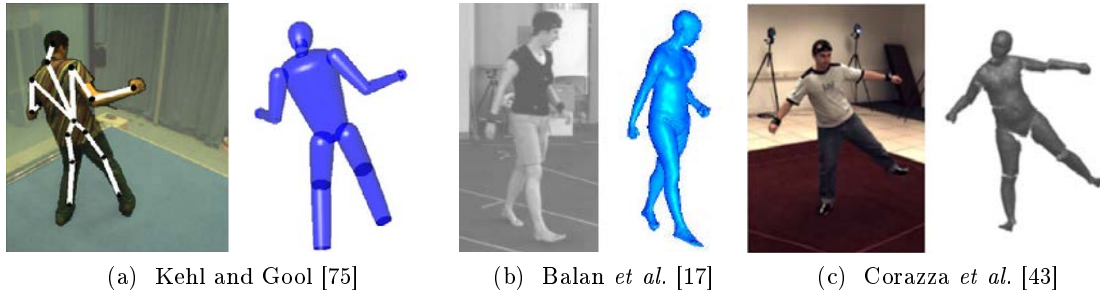


FIGURE 2.10 – Examples of model-based methods. Left: video picture of the captured pose. Right: reconstruction by model fitting.

of visual hulls. A *Reeb graph* [54, 159] is extracted in each pose. The frames are then segmented in groups of topologically identical graphs. Articulated skeletons are extracted with a motion-based clustering of the graphs. Next, A unique consensus kinematic structure is fitted with the successive skeletons. This method results in the tracking of a skeleton only. Another type of skeleton-based 3D pose tracking is achieved by Ukita and al. [160]. The pose of a human body is estimated from a visual hull series, using pose regression [155]. This method involves an offline learning step with a set of refined body volumes and their associated skeleton pose.

2.5.2.2 Parametric human models

Plankers and Fua [125] use an articulated template made of a set of 230 *metaballs* (or *soft objects*) [51] attached to a skeleton. A set of implicit surfaces (for each limb) are computed from these metaballs whereas explicit surfaces are used for the head, hands and feet. The shape of this model is then estimated from video sequences by fitting in each frame a disparity map, obtained by stereo matching, and a silhouette constraint. Carranza *et al.* [34] described a marker-free motion capture which employs an articulated generic human body model initialized using the silhouette images that show the actor in an initialization pose. The tracking throughout the video sequences is performed afterwards by finding the model pose parameters which maximize the overlap between projected model silhouettes and the silhouettes of the actor captured by the cameras. Kehl and Gool [75] proposed a method to follow the movements of an actor using a discrete volume obtained by a visual hull reconstruction (see Figure 2.10a). A basic human model, composed by *superellipsoid* limbs, is linked to a skeleton. This articulated body is then matched with 3D (volumetric reconstruction) and 2D cues (multi-view images edges) through a Stochastic Meta Descent (SMD). A similar approach is described by Mikić *et al.* [109] who used a parametric model, representing a simplified human boyd, to track a sequence of volumetric multi-view reconstruction through an *Extended Kalman Filter* (EKF). Horaud *et al.* [64] use an articulated model of 21 ellipsoids. An implicit surface is defined as a blending of these ellipsoids. This articulated implicit surface is then fitted to a set of 3D points, computed from a visual hull reconstruction, through an *Expectation-Maximization* (EM) algorithm.

2.5.2.3 Hybrid models

To reconstruct the actor's shape with a higher fidelity, a pure articulated template model is not adequate, a free-form surface is required in addition. One of the earliest method to capture skin deformation was described by Sand *et al.* [132]. This approach is a mix of traditional Mocap and multi-view reconstruction. A skeleton is first fitted with the actor's poses by a marker-based motion capture process. A set of segments (*needles*) are rigidly and perpendicularly attached to the skeleton's bones. The intersection between these needles and the multi-viewpoint silhouettes define a sampling of the skin surface that is then modeled by a set of deformable primitives. The human body model contains several of these primitive for each limb. This method is still limited by the usual motion capture inconvenient (the actor wears specific neutral clothes and markers) and the reconstruction is thus restricted to this specific acquisition setup. De Aguiar *et al.* [46] use a skeleton-based model with body segments composed by b-splines surfaces. After a global articulated deformation to fit the silhouettes, the surface is locally deformed to match the photo-inconsistent parts of the body's geometry. Balan *et al.* [17] use the SCAPE model (*Shape Completion and Animation of People* [13]) as a template (see Figure 2.10b). This is a parametric human model, synthesized from scanner acquisition of several subjects, which includes both articulated and non-rigid deformations. This model can be used for marker-based motion capture but is here animated by matching, for each frame, the mesh projections towards multi-view cameras with silhouette pictures of the actor. Zhang *et al.* [174] used a model-based approach for marker-less facial reconstruction. Their method tracks a human face template over point cloud series, obtained from structured light and multiple viewpoints capture. Here, the use of a template shape is also a way to deal with gaps in point cloud capture and to obtain a full reconstruction of the face during the whole capture.

2.5.2.4 Subject-specific models

The more recent methods usually build a template model from a direct acquisition of the actor. The template is then more adapted to the morphology of the tracked character and also gives more realistic results. These approaches still often rely on an articulated template. To automatically attach a template skeleton to a mesh, several methods have been developed. The algorithms proposed by Baran and Popović [18] and Tadano *et al.* [148] (which uses Reeb graph [54]) are widely used.

The Marker-less Motion Capture method by Corazza *et al.* [42] takes as template a laser scan of the subject, manually segmented into a kinematic model, to track visual hull point cloud series through a stochastic approach (*simulated annealing*). Follow-up work [43] uses a *subject-specific model* from a laser scan (or a visual hull reconstruction) of the actor (see Figure 2.10c). This model is tracked over a time series of visual hulls by a kind of *articulated ICP* method [49, 113]. Luo *et al.* [102] construct their template by a visual hull reconstruction. Next, this model is segmented and fitted to a skeleton using a modified version of the Tadano *et al.* algorithm. The human motion is then captured by tracking a sequence of visual hulls. Finally, the model's surface is deformed according to a silhouette constraint to fit the captured data. Kilner *et al.* [76] perform an action matching for sport broadcasting. Their motion tracking is applied on outdoor sportive scenes. Instead of traditional 2D pose estimation technique from a single camera, this

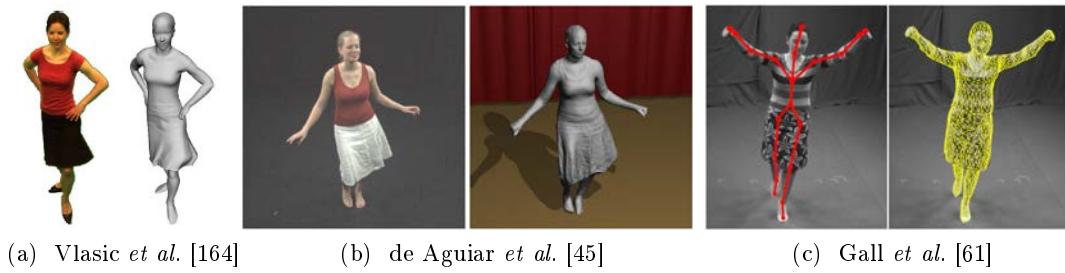


FIGURE 2.11 – Examples of model-based methods. Left: video picture of the captured pose. Right: reconstruction by model fitting.

matching is performed by comparing a 3D reconstruction with a 3D library of human key poses, using *volumetric shape histograms* [67]. Even if this approach deals with very small silhouettes and therefore lacks of precision, this method is one of the few dealing with multiple actors' reconstruction. The initial 3D reconstruction is performed by *Conservative Visual Hull* (CVH) [77] that is similar to usual visual-hull reconstruction, except that the silhouette masks are expanded by n pixels to avoid truncation involved to calibration errors (even if this operation may introduce phantom volumes).

2.5.3 Realistic model-based methods

Derived from the methods described in the previous section, a lot of more recent temporally consistent multi-view reconstructions use a model-based approach. However, instead of a simple motion tracking, these techniques have to reconstruct the shape of the subject in a realistic way.

2.5.3.1 Subject-specific articulated models

A lot of methods use a full body scanner to obtain a detailed 3D mesh of the actor. De Aguiar *et al.* [47] compute an automatic segmentation of this mesh, based on a convexity criterion, to get a regular sampling of the surface. The centers of these clusters are used as markers and connected to build a graph. They also compute the optical flows which describe the motions over the multi-view video sequences. The markers are animated following this displacement flow, and, this way, drive the motion of the graph. The whole mesh is finally deformed, following the graph through a Laplacian deformation framework. As we noticed in the previous section, the purely articulated models (skeletons or other types of control structures) are unable to reconstruct a realistic sequence, especially when they contain free-form surfaces such as loose clothes. Theobalt *et al.* [156] and Ahmed *et al.* [2] use a kinematic template model with smooth and neutral surface and add on this surface thin details (*e.g.*, wrinkles) captured by a shape-from-shading approach. However, the global shape of the reconstruction is still strongly limited by the template model. Anguelov *et al.* [12] use a probabilistic model for the registration of non-rigid 3D surfaces. A template detailed mesh is deformed to match and complete the data, made of partial scans, without making assumptions about object shape or dynamics. Vlasic *et al.* [164] use a scanned articulated template mesh as input (see Figure 2.11a). A skeleton is automatically fitted to each frame of a visual hull reconstruction (a user intervention may be necessary for

uncertain cases due to complex poses or occultations. The template is deformed by a *Linear Blend Skinning* (LBS) [89] algorithm to match the new pose of the skeleton. At last, local deformations allow to closely fit the silhouettes. Like in [47], these deformation are based on a Laplacian coordinates conservation framework that ensure the preservation of the surface details. Gall *et al.* [61] use a similar approach, except that the deformation of the template is guided by photometric constraints (2D silhouettes extracted from the multi-view videos) rather than a 3D reconstruction (see Figure 2.11c).

2.5.3.2 Non-articulated models

Most of the recent approaches are based on such a mix of articulated pose-matching and local free-form deformations. Nevertheless, some authors developed model-based approaches that use less constraining deformation process. Duveau *et al.* [56] use a *cage-based animation* [95] instead of a skeleton-based method to track a time series of visual hulls. In a relevant method [45], de Aguiar *et al.* transform the scanned template mesh into a tetrahedral mesh animated by a *volumetric deformation* (see Figure 2.11b). This procedure is based on a linear Laplacian deformation, like the *As-Rigid-As-Possible* deformation [138] except that it is applied on the tetrahedral structure instead of a triangulated mesh. This approach seems close to [26]. The motion of the actor is estimated in the multi-view videos with a feature matching based on the SIFT algorithm [99]. The tetrahedral mesh is then deformed according to this point-based constraints. Finally, the vertices are locally displaced to match the silhouette pictures and stereo depth-maps extracted from each viewpoint. Cagniard *et al.* [33] and follow-up work by Allain *et al.* [7] employ a dynamic surface, based on a deformable tile set, initialized with the first frame of the sequence. This mesh is then deformed to fit the subsequent poses. For each pose, the patches are matched to the corresponding part of the surface in the next frame according to a distance function between the two surfaces. The authors also propose a volumetric approach [6] based on *Centroidal Voronoi Tessellations* (CVT) [55] which cells are similarly clustered in rigid patches. This template is fitted with the data through a probabilistic approach, following an EM process [50].

The main advantage of this family of techniques is the temporally-coherent animation they produce and the visual quality of the result. Nevertheless, the use of a template model restricts their generality. Model-based methods are often limited to a single human model, even if some variants allow the tracking of several actors. For instance, Liu *et al.* [97] extended the method of Gall *et al.* [61] to track two characters in the same scene. However, assuming a specific template model like in this first family of methods is often too restrictive to capture arbitrary motion sequences: for instance, skeleton-based approaches lead to strong limitations when applied to actors wearing loose costumes (dresses, coats) or accessories (bags, hats) if no extra local optimization is performed.

Nonetheless, several of these techniques, despite the visual quality of the results (due to the high precision scanner reconstruction of the template) roughly matches the actor's pose but with a low adaptation to the tracked surface, such as the clothes' deformations (*e.g.*, [45]). The reconstruction approaches based on an articulated model representing a generic human body can be considered closer to a marker-less motion capture technology. These methods are not well suited for film production because the results are poorly realistic. However, the visual quality of these approaches can be sensitively improved by using a template model built from the filmed actor (for example by making a 3D scan acquisition

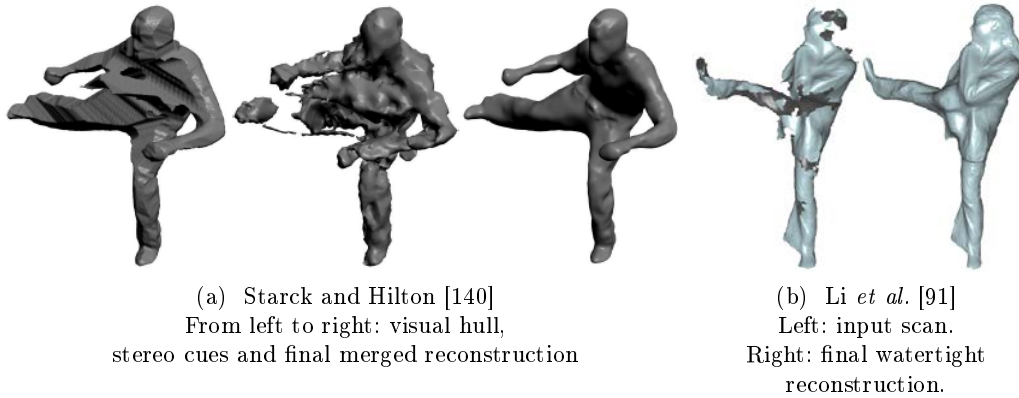


FIGURE 2.12 – Examples of model-free methods.

of the actor before the shooting). Therefore, this requires a heavier preliminary processing and reduces the genericity of the system.

2.5.3.3 Point-cloud sequences completion

Many methods based on scanned point-cloud series also fit a template geometry to ensure a consistent reconstruction and fill the holes that appear in this kind of reconstruction [30]. For example, one of the earliest method by Allen *et al.* [9] fits a skeleton which controls a template surface to compute correspondences between different range scans. In follow-up work [10], this articulated prior is enhanced by local deformations, similar to [144]. Li *et al.* [90] compute a non-articulated template model by a static subject-specific acquisition. This template is deformed through a non-rigid ICP algorithm to match and fill a series of partial scans, reconstructed by a real-time structured light stereo reconstruction (see Figure 2.12b). The thin details of the surface are aggregated over the animation, even if they only appear on a few frames. The main limitation of the model-based methods are that the details of the surface are limited to those contained in the template. The more realistic methods give convincing result due to actor specific template acquisition and local optimization step. Moreover, these templates often contain richer details than the input reconstructed poses. It is thus a way to produce a fine and detailed animation from a low resolution reconstruction, but it may lead to inconsistencies between the final animation and the original captured data. It can then be seen as marker-less motion capture, rather than a real reconstruction of the observed scene. However, the free-form motion tracking remains limited. Another general problem is the inability to deal with changing topology. In the case of such changes in the reconstructed time series (*i.e.*, due to collisions), they are often considered as reconstruction artifacts, the topology of the actor being not supposed to change during the animation.

2.5.4 Temporally consistent model-free methods

Even if the temporally coherent reconstruction is usually associated with model-based approaches, some techniques allow to compute a time-consistent mesh without a predefined template shape.

2.5.4.1 Dynamic surface from mesh sequences

Starck and Hilton [140] proposed a model-free method based on visual hull and stereo reconstruction, enhanced by feature-matching through a graph-cut [29] process. A spherical parameterization is then operated on the object, following the method described by Braun and Hoppe [128] (see Figure 2.12a). This restricts the process to work only on single closed surfaces. The authors bypass this problem by using a mesh cutting operation [142]. The inter-frame consistency is ensured *via* this spherical domain embedding, by resampling the mesh on a uniform domain [177]. The whole process is still sensitive to complex surfaces with narrow extremities.

2.5.4.2 Articulated tracking of point-cloud sequences

Zheng *et al.* [176] developed a method to transform a time series of point clouds, acquired by a structured light scanner, into a unique animated object. They first extract a skeleton from each frame of a scanned sequence. They then compute a unique consensus skeleton matching the successive poses, to derive a time-consistent reconstruction (following the same kind of approach as Mukasa *et al.* [112]). Similarly, Pekelny and Gotsman [122] accumulate a time series of partial scans, taken by a single depth video camera, to compute an optimal articulated object. The points of the whole reconstructed object are segmented and skinned to a skeleton (this requires a manual segmentation of the first frame and specification of the skeleton connectivity). Nevertheless, these approaches are limited to clearly articulated shapes, which is not compatible with our goal. Chang and Zwicker [36] propose another registration of range scans of deforming shapes, but their method is also limited to subjects which presents articulated motions only.

2.5.4.3 Non-articulated tracking of point-cloud sequences

Mitra *et al.* [110] proposed a method for dynamic registration of scanned surfaces by computing rigid transformations. A *space-time surface* is computed by the accumulation of the points cloud acquisition in a single four dimensional space ($3D + t$). Next, a normal vector is estimated in each point of this surface. The rigid motion between two poses should be perpendicular to the normal field (*kinematic constraint*). Therefore, the velocity vector between consecutive frames is computed by minimizing the difference between this vector and the tangential planes of each point. The authors then propose an extension for deformable bodies, noticing that their transformations can be considered as locally rigid. However, their methods seems sensitive to fast motions and important inter-frame deformations. Sükmuth *et al.* [145] describe a similar approach where the point clouds captured by a fast 3D scanner are accumulated in a single time-space surface. This surface is approximated by a single 4D implicit function which first time-step is used to compute a template polygonal mesh. This mesh is then deformed along the time axis with an as-rigid-as-possible deformation [138].

Wand *et al.* [168] compute a dynamic shape and its deformation from fast scanner series, using a statistical framework. The transformation is based on a geometric alignment of adjacent scans (using a variant of non-rigid ICP [62]), with a temporal smoothness constraint, *via* a global optimization scheme. A more recent approach [167] employs a

subspace deformation technique to compute a complete surface model from partial input data. However, it is still sensitive to large time steps. Popa *et al.* [126] describe a template-free reconstruction to iteratively compute correspondences between adjacent frames, using optical flows as hint, and produce temporally local consistent frame sequences until they get a final animation over the whole time series. However, these techniques cannot handle fast motions and are still sensitive to large time steps. Tevs *et al.* [153] match a set of landmarks correspondences that are used to compute a dense mapping over a scanned sequence. The accumulated partial scans then results in an animated model. We can also mention Li *et al.* [91] who developed a temporally consistent completion of scanned meshes' sequences, using a deformation graph to establish pairwise correspondences between adjacent frames (see Figure 2.12b). This registration integrates a template-free, non-rigid ICP algorithm, extended from [90]. Nevertheless, this method is applied to high definition meshes (despite the scan holes) whereas our visual hull surfaces are less trustworthy.

Bonarrigo *et al.* [25] use a patch-based approach to compute the registration of scanned point-cloud series, with large motion and, thus, a low overlapping between the successive frames. The scanned surfaces are sampled as a set of nodes that define partially overlapping patches. A rigid transformation is computed for each patch by minimizing an objective function. A fitting term matches the nodes of the two consecutive surfaces, according to the nodes' radius of influence, whereas a regularization term penalizes the difference between neighboring nodes' transformations. The transformation of each point of the surface is finally interpolated from the nodes' displacements, leading to an as-rigid-as-possible deformation.

2.6 Conclusion

As a conclusion, we first discuss the various approaches presented in this chapter (see Section 2.6.1) and then present an overview of the approach we propose (see Section 2.6.2).

2.6.1 Discussion

Our goal is to get a temporally coherent representation of a 3D scene from a sequence of time-varying 3D objects made by a silhouette-based multi-view reconstruction. In this case, several model-based approaches seem relevant. However, the model-free reconstruction methods (especially shape-from-silhouette and shape from-stereo) are the most generic and present the advantage of being able to reconstruct any type of scene without assumption about its content or preprocessing specific to the actor. They are also less complex and can be processed in a reduced computing time. For all these reasons, a model-free reconstruction can be used to compute an initial sequence of reconstructions which can then be considered as input data for a pose-tracking approach. In RECOVER 3D, a silhouette-based reconstruction is used for real-time reconstruction of the actor. The resulting sequence of rigid poses is then used as input to compute a time-consistent animation. Many state-of-the-art techniques are based on a motion tracking of a pose sequence, using a template body. Most of these methods use an articulated shape and a predefined template surface (see Table 2.1). This requires a generic skeleton and human shape as input. For a realistic rendering, this shape must be constructed from a prerequisite reconstruction of the filmed actor. These constraining inputs are a major drawback in our framework. Indeed, we aim

at developing a generic process that should deal with various types of characters. Since our application domain is TV production content, these characters could wear costumes and accessories. Future application may also include non-human bodies (animals for instance). In most of these situations, a skeleton fitting is challenging, and inappropriate to animate the captured surface (with a dress for example), even if these skeleton-based reconstruction often include a non-articulated surface optimization. A free-form tracking approach will be necessary. However, several of these techniques suffer from a high complexity. Moreover, most of template-free mesh registration methods are limited to small displacements. In the case of multi-view reconstruction of real actors, captured by video cameras only, the velocity of the motions and the reduced frame-rate can lead to large displacements between two consecutive poses. The deformation of a surface from one pose to another thus should have to be driven by a motion information. This motion can be extracted from the input information and then applied to the mesh through an adapted deformation prior.

2.6.2 Our approach

Our approach is inspired from all these different classes of methods while resolving some of their limitations. As in model-based approaches we deform a template mesh in order to match the captured time series of visual hulls. Our objective is to track the surface of these successive poses. In this case, most of the state-of-the-art methods use a model-based approach (see Table 2.1). However, we seek for a generic process and thus cannot be limited by the morphology of a fixed template. Our production framework also cannot produce a subject-specific template like the most robust approaches described in Sections 2.5.2.4 and 2.5.3. Accordingly, our template surface will be the mesh reconstructed from the first frame of the sequence, so that we do not need an input model nor an initial pose. Most of model-based techniques use articulated templates, which are not adapted to our objectives (see Section 1.5). Instead, our approach is based on a non-rigid tracking (see Section 2.5.3.2). The template mesh is taken from the initial model-free reconstruction to avoid a complex pre-processing operation of template building which could be time-consuming and/or require specific hardware. Our approach could be compared to the non-rigid tracking approaches described in Section 2.5.4, except that these methods are often adapted to high-frequency 3D scans and therefore sensitive to large displacements. Our surface tracking will thus be guided by a preliminary motion estimation to drive the global deformation of the template, as in several model-based approaches (see Sections 2.5.2 and 2.5.3), while maintaining the non-articulated prior. Motion flows could be considered limited due to the amplitude of the movements but they can help managing any type of model and work on volumes. The motion vectors drive the global deformation of the template shape more efficiently than an approach only based on a pose convergence which could be sensitive to large displacements or remain stuck in local minima. This pseudo rigid deformation thus enables both a free-form matching and a local rigidity. At last, a final optimization step may be necessary to deal with the complex cases such as local variations of the surface or reconstruction artifacts, as proposed by most of the hybrid model or free-form approaches (see Sections 2.5.2.3 and 2.5.4.3).

Chapter

3

Motion extraction

Contents

3.1	Introduction	55
3.2	Positioning in the context of previous work	55
3.2.1	Motion estimation	55
3.2.2	3D motion tracking	56
3.2.3	Our approach	57
3.3	Input sequence representation	58
3.3.1	Volume series	59
3.3.2	Distance volume coding	59
3.4	Voxel matching	60
3.4.1	Proximity criterion	61
3.4.2	Orientation criterion	61
3.4.3	Colorimetric criterion	61
3.5	Motion flow regularization	62
3.6	Implementation and improvements	63
3.6.1	Limitation of multiple matchings	63
3.6.2	Prediction	63
3.7	Results	64
3.7.1	Evaluation of the motion flow reconstruction	65
3.7.2	Discussion on the chosen parameters	66
3.7.3	Mesh animation	67
3.7.4	Limitations	69
3.8	Conclusion	71

Résumé

Nous décrivons dans ce chapitre la première étape de notre méthode de suivi de poses qui consiste à calculer un champ de déplacements décrivant les mouvements de l'acteur. Nous utilisons comme données d'entrée les séquences de volumes discrets obtenus après une reconstruction basée silhouette. Ces volumes sont représentés sous formes de grilles 3D régulières de *voxels* (*Volume Elements*). Chaque voxel correspond à une valeur qui définit s'il est à l'intérieur ou à l'extérieur de l'objet. Une transformée en distance euclidienne (*EDT*) est aussi préalablement calculée pour chaque volume. L'*EDT* est une grille de même résolution que le volume où chaque voxel est défini par une valeur correspondant à la distance euclidienne au carré entre le voxel et la surface (zone de contact entre voxels intérieurs et extérieurs) la plus proche.

Afin d'estimer la trajectoire des mouvements effectués entre deux poses successives de la séquence, nous commençons par calculer un appariement entre les voxels des deux volumes correspondants. Soit deux volumes V^n et V^{n+1} consécutifs correspondants aux poses n et $n+1$ d'une série de N volumes. On souhaite réaliser un appariement $V^n \rightarrow V^{n+1}$. Commençons par définir comme *voxels de surface* les voxels intérieurs qui sont en contact avec au moins un voxel extérieur. Ces voxels de surface sont caractérisés par une couleur au format RGB (texture issue des image multi-vues) et un vecteur normal à la surface en ce point. On cherche à associer chaque voxel de surface $v_i^n \in V^n$ au voxel de surface $v_j^{n+1} \in V^{n+1}$ qui lui correspond le mieux. On juge que l'appariement de deux voxels est satisfaisant lorsqu'il minimise la fonction distance décrite par l'équation 3.3. Cette expression est une somme pondérée de trois critères : proximité (*cf.* Section 3.4.1), orientation (*cf.* Section 3.4.2) et colorimétrie (*cf.* Section 3.4.3).

Pour chaque voxel de surface v_i^n , on parcourt les voxels de surface de V^{n+1} qui se trouvent dans un voisinage fixé (*cf.* Section 3.5) et l'on retient le voxel v_j^{n+1} qui correspond au plus petit résultat renvoyé par la fonction de distance. Le vecteur défini par la position des deux voxels v_i^n et v_j^{n+1} est ajouté au champ de vecteurs en v_i^n . Ce champ de vecteurs a la même structure que la grille de voxels. Chaque position peut contenir un ou plusieurs vecteurs. La même opération est répétée une seconde fois, en cherchant cette fois pour chaque v_j^{n+1} le voxel de surface v_i^n le plus proche. Les vecteurs retenus sont à leur tour ajoutés au champ de vecteurs en v_i^n . Ce second passage permet de retrouver une partie du mouvement qui pourrait avoir été ignoré lors du premier appariement. On s'assure ainsi que chaque voxel de surface de V^n et V^{n+1} est associé à au moins un vecteur.

L'étape précédente nous permet de récupérer un champ de vecteurs initial qui décrit le mouvement de l'objet entre V^n et V^{n+1} . Cependant de nombreux appariements peuvent être erronés et cet ensemble de vecteurs est trop irrégulier pour être exploitable. Une opération de *lissage* est donc ensuite effectuée en appliquant un filtre gaussien 3D sur le champ de vecteurs initial. Le résultat final est un champ de vecteurs qui donne une estimation du déplacement de la surface entre deux poses consécutives. l'opération est répétée pour chaque couple de poses (V^n, V^{n+1}) tout au long de la séquence (de $n = 0$ à $n = N - 1$) pour obtenir une description des mouvements sur l'ensemble de l'animation.

3.1 Introduction

This chapter describes the first step of our method. We first give an overview of 3D motion estimation principle in Section 3.2. Our approach is detailed in Section 3.3 to 3.6. Section 3.7 presents preliminary results and outlines a motion flow-driven mesh animation approach.

3.2 Positioning in the context of previous work

Our goal is to get an estimation of the motions that happen in a 3D scene reconstructed from a moving actor’s performance. An initial model-free technique performed a volumetric reconstruction from each frame of the initial multi-viewpoint video streams. Given that our application should be used as an offline, post-production tool, our input data are the reconstructed sequences alone. Our approach does not use the multi-view pictures from the original videos. Indeed, we thus avoid dealing with voluminous data that may not have been stored after the initial reconstruction process or could be hardly transferred to the system. This overview provides an introduction to motion estimation (see Section 3.2.1) and a brief review of previous work in 3D motion tracking (see Section 3.2.2). We introduce our approach in Section 3.2.3.

3.2.1 Motion estimation

The estimation of the motion that appears in a captured scene is a widely studied problem. The most common techniques compute an *optical flow* between the consecutive pictures of a video, throughout the whole sequence. An optical flow is a set of 2D vectors, one for each pixel, that represent the displacement of the corresponding point as seen from the camera viewpoint. Among the various approaches to compute the optical flow, two noteworthy methods have been described by Lucas and Kanade [101] and Horn and Schunck [65]. They both rely on the principle that, in a stable environment with a constant lighting, the intensity of a captured point remains unchanged between two consecutive snapshots. This leads to the *brightness constancy constraint* formulated by the following equation:

$$I_x V_u + I_y V_v = -I_t \quad (3.1)$$

where I_x , I_y and I_t are the partial derivatives of the image I along the spatial axis x and y and time t , respectively. $V(u, v)$ is the velocity vector which defines the displacement between two images. The equation 3.1 can be rewritten in the matrix form:

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \begin{bmatrix} V_u \\ V_v \end{bmatrix} = -I_t$$

which becomes:

$$\nabla I^T \cdot \vec{V} = -I_t \quad (3.2)$$

where ∇I is the *spatial derivative*, or *gradient*, of I . This equation contains two unknowns and its solving is therefore challenging. This is linked with the *aperture problem* of the optical flow algorithms: the motion of an edge seen through an aperture is ambiguous and may be consistent with many different motions. All optical flow computation methods thus

introduce additional constraints to find a solution. The **Lucas-Kanade** [101] approach is a differential method that relies on the assumption that the velocity is locally constant. For each pixel q_i , a local equation system, built by applying the equation 3.1 for each pixel in a fixed window around q_i , is solved by a least-square approach. The **Horn-Schunck** [65] method assumes a global smoothness of the flow through the whole image. The flow is formulated as a global energy to minimize.

These approaches are often sensitive to large displacements that could occur between consecutive frames. Other types of methods are based on a *block-matching* approach [93, 178, 115]. This consists in tracking the point between two adjacent frames by searching, for each pixel at a fixed time t , for a similar pixel in the next frame $t + 1$. This principle is similar to a feature matching algorithm. A pixel and its close neighborhood form a *block*. The algorithm looks for a block with the same properties in the next picture and then matches the pixels at the center of these two blocks together. The block matching algorithms are, for example, widely used in video compression. It should also be noticed that the optical flow does not describe a 3D motion but 2D motion as seen from the capture viewpoint, *i.e.*, the 3D displacement projected toward the camera.

3.2.2 3D motion tracking

These techniques described in Section 3.2.1 can be applied to 3D motion recovery in a multi-camera context, with multiple viewpoints. In a relevant article [163], Vedula *et al.* defined the concept of *scene flow* by the 3D equivalent of the optical flow. The optical flow describes an instantaneous motion field in an image. In the same way, the scene flow is a three-dimensional flow field which describes the motion at every point in the scene. Vedula and coworkers compute the 3D motions vectors of this scene flow, in the real scene space, by merging the optical flows computed from multiple viewpoints around the stage. In the context of multi-view reconstruction, a motion flow computation can be considered by working on the poses of a reconstructed sequence of static 3D objects. This representation of the animated scene can be compared to a 3D equivalent of a video as it is composed of static snapshots without registration of the movements. Many scene flow computation technologies are based on stereo-based reconstruction and depth-maps [170, 86, 166] or surface patches (*surfel*) photo-consistency in multi-viewpoint environments [114, 70].

Several motion-tracking approaches have been proposed to achieve a reconstruction with temporal consistency from time series of static objects that typically results from model-free reconstructions. Various types of methods perform a *mesh-tracking*, applied on meshes obtained through multi-view reconstruction. Several meshes can be matched according to curvature, texture criteria, photo-consistency, or various types of features [173] from which one can compute the motion flow describing the movements of an actor between two frames [124]. These methods first match two meshes by tracking 3D features on the reconstructed surfaces, eventually mixed with 2D features from the multi-viewpoint videos. These sparse correspondences are then propagated through the whole surface to finally obtain a dense matching. Starck and Hilton [139] track surface point features, using a set of edges, corners, and region descriptors. The correspondences for all vertices are computed through MRF energy minimization afterwards. Tung and Matsuyama [158] propose a similar algorithm, with an initial correspondence matching based on geodesic distances of the vertices. Varanasi *et al.* [162] mix image and mesh features, using SURF descriptor and geodesic integral, respectively. The motion field is obtained through Laplacian dif-

fusion. Furukawa and Ponce [60] deal with meshes reconstructed from multi-view stereo acquisition. They use photometric photo-consistency to compute the local motion of the vertices, regularized according to the mesh’s adjacency.

Our meshes are provided by a volumetric silhouette-based reconstruction which produces digital volumes sequences, that are then transformed into meshes with a surface extraction algorithm. In our case, however, the visual hull reconstruction usually creates significant artifacts (*e.g.*, phantom volumes) which prevents us from using such a mesh-tracking approach directly. Therefore, the surface of the same object in different frames may highly differ, both in topology and surface details. This way, a mesh tracking approach using mesh features or surface properties may fail to establish a robust matching. Instead, we use a volumetric approach to compute a motion flow based on a voxel matching algorithm applied to the input sequence. Anuar and Guskov [14] proposed to compute a 3D optical flow, based on a hierarchical adaptation of the Lucas-Kanade approach, on a voxel grid. A distance transform is used as a 3D image, with distance values instead of intensities. The motion flow can then be used to animate a mesh over time. Nobuhara and Matsuyama [117] use a *voxel-matching* algorithm between the successive poses of a volumetric visual hull sequence. This algorithm establishes correspondence according to Euclidean distances. A template is obtained by a *marching cubes* triangulation of the first frame volume and then animated following the displacement vectors. However, the motion flows computed in this method are simply obtained by matching each voxel to the closest one in another frame, thus producing motion vectors which lack accuracy. Destelle *et al.* [53] describe a motion flow computation from point clouds, using a multiresolution voxel grid. The voxels from each of the two poses are associated with a voxel from the opposite pose, in a *back-and-forth* matching based on a distance function which includes a Euclidean distance and a normal vector angular difference. This algorithm penalizes the matching of voxels from surfaces which do not have the same orientation. This raw motion flow is then filtered by computing the gradient of the vector field and removing the outliers according to a fixed threshold.

3.2.3 Our approach

As described in Chapter 1, we take as input sequences of visual hulls obtained through a volumetric silhouette-based reconstruction (see Section 2.4). We assume that the input is a binary 3D array which represents a sequence of poses generated by this reconstruction process along with the colors captured by the video. These volumes are usually transformed into a sequence of 3D textured meshes, successively loaded for the rendering of each frame. In this constrained industrial framework, our goal is to introduce a dynamic representation of the captured character, that adds a temporally consistent description of the scene. Our ultimate goal is to generate a single, time-evolving triangle mesh representing the motion of the actor in the capture sequence. This technology should be used to perform the reconstruction of various types of scenes, involving actors wearing costumes and accessories, or even animals. This requirement prevents the use of most existing methods which assume rigidly articulated models. This project is oriented towards broadcast and TV markets, that is why rapid computations are favored to match the constraints of short production deadlines.

To answer these requirements, we developed a new method which uses a feature-based volume tracking to identify the actor’s motions and then apply a surface matching algo-

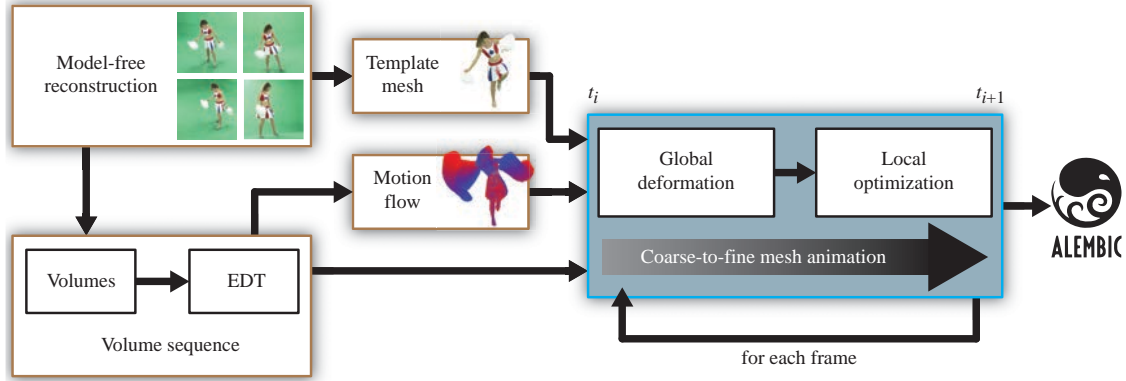


FIGURE 3.1 – **General process flow.** After a model-free reconstruction, we compute a volume sequence and a template mesh. A motion flow is computed from the volume sequence (Section 3.4). The template mesh is then animated, according to this motion flow, by a two-step mesh deformation (see Chapter 4). The animated mesh is finally saved into a file before being used in a post-production software.

rithm. The different steps are illustrated in Figure 3.1. We extract the scene motion by computing a 3D motion flow from the input 3D volume sequence. Our method relies on a voxel-matching algorithm that establishes correspondences between adjacent poses of the reconstructed time-series to get an estimation of 3D vectors (see Section 3.4). The particularity of our method is to combine complementary criteria (proximity, orientation and color) that allow to discriminate voxel matching, with a back and forth approach. This method should work on generic datasets, whatever the shape of the reconstructed object or character. Our input is a sequence of n digital volumes. The i -th volume of this sequence represents the actor at time t_i . This input data is described in Section 3.3. This initial set of trajectories are then regularized to finally obtain a motion flow which describes the movements of the actors between two poses (see Section 3.5). In the next step we use this flow to animate a dynamic mesh model. The reconstructed mesh at the first frame, obtained by a *marching cubes* algorithm, is used as the initial template model. A coarse-to-fine process explained in Chapter 4 is iterated between pairs of frames until the end of the sequence to consider. As show in Figure 3.1, the two-step process (motion flow extraction and template mesh deformation) is iterated over the complete sequence (N poses), for each successive pair of poses (t_i and t_{i+1}), until the end of the animation ($i \in [0, N - 1]$). Traditional articulated model-based approaches are too restrictive for our application context. We thus use a non-rigid mesh deformation which allows free-motion tracking which is significantly less restrictive. By deforming the mesh at each frame according to the estimated flows to match the subsequent captured frames, we deduce a character’s animation.

3.3 Input sequence representation

For each frame of the synchronized multi-view videos, a volumetric visual hull is reconstructed. Next, we compute a distance transform on this input. Therefore, our data are sequences of visual hulls which each pose is represented by both a discrete volume (see sequence 3.3.1) and a distance grid (see Section 3.3.2).

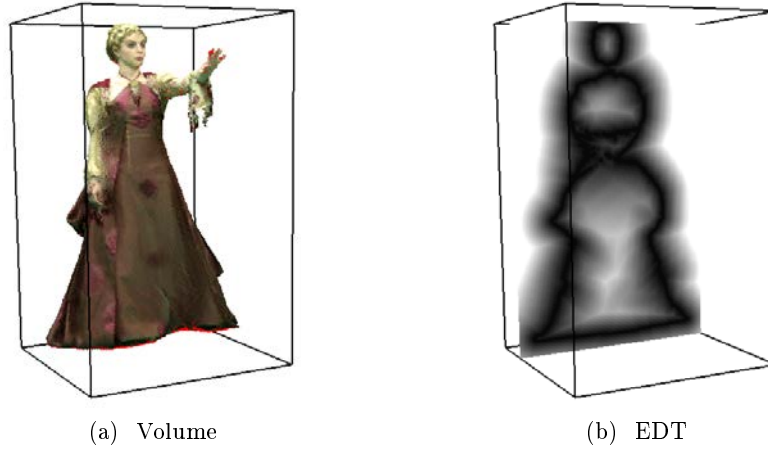


FIGURE 3.2 – **Volume processing.** (a) An example of colored reconstructed volume issued from multi-view reconstruction. (b) A sliced representation of the corresponding *EDT*.

3.3.1 Volume series

The reconstructed volumes we use as input are simple digital volumes, represented by a regular 3D grid of isotropic binary voxels (0 for void voxels and 1 for voxels covering or intersecting the object). Voxels straddling the surface (*i.e.*, those that are not void, yet direct neighbors of void voxels) are assigned a color extracted from the multi-view video frames: the color associated to a voxel is interpolated from the multi-viewpoint images which contains this point. Each surface voxel is then associated to a RGB color (see Figure 3.2a). Note that for simplicity of our data structure, we implement a volume as an RGBA array, where the alpha channel is set to 0 for void voxels, 1 for internal voxels, and 0.5 for surface voxels.

3.3.2 Distance volume coding

We then compute another representation of these volumes by using a *Euclidean distance transform* (EDT), as described by Saito and Toriwaki [131]. This transform is computed by applying a forward and backward path to compute, for each point, the minimum distance to the closest surface point. This two-step operation is repeated for each axis of the 3D grid to update the distance values, following the generalized Pythagorean theorem (the square of the distances on each axis equals the squared Euclidean distance). At the end of the process, each voxel is assigned to the smallest squared distance to the closest surface voxel. The algorithm is given in appendix B.1. We obtain an unsigned distance volume, represented by a 3D grey-level voxel grid, as shown in Figure 3.2b. Each voxel is assigned to a positive value which corresponds to the Euclidean distance to the closest boundary of the object. This volume description can be considered as a grey-level 3D picture. Thus, we are able to compute a derivative estimation of this picture. It will be used to compute the normal vectors (see Section 3.4.2) and gradient values. To compute the spatial derivative, we use a set of Sobel-like filters which estimate, in a 3^3 window around each voxel, the EDT variations for each spatial axis. A temporal derivative is also computed on the same neighborhood by the differences of the values between two consecutive frames.

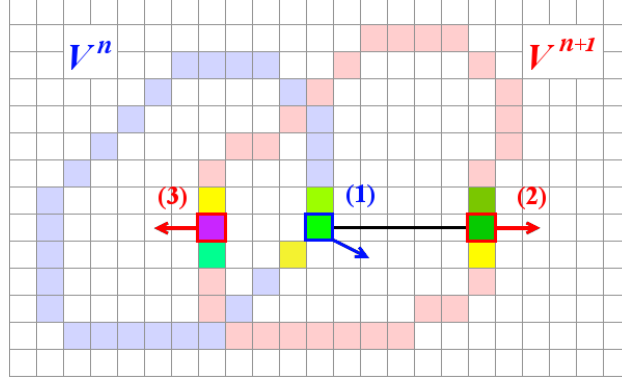


FIGURE 3.3 – Voxel matching between two consecutive volumes. The voxel (1) from the V^n volume matches better the voxel (2) from the V^{n+1} volume than the voxel (3). The neighboring voxels are represented with their colors. Normal vectors are depicted by arrows.

3.4 Voxel matching

Given two adjacent volumes V^n and V^{n+1} which correspond to frames t_n and t_{n+1} , our goal is to compute a matching $V^n \rightarrow V^{n+1}$ representing the scene flow. We define as *surface voxels* the voxels which belong to the object and have at least one void voxel in their direct neighborhood. These surface voxels are characterized by an RGB color and a surface's normal vector. We want to match each surface voxel $v_i^n \in V^n$ to another surface voxel $v_j^{n+1} \in V^{n+1}$ minimizing the following distance function:

$$D(v_i^n, v_j^{n+1}) = \omega_p \delta_{i,j} + \omega_n \varphi_{i,j} + \omega_c \sigma_{i,j} \quad (3.3)$$

where $\delta_{i,j}$, $\varphi_{i,j}$ and $\sigma_{i,j}$ are normalized and correspond respectively to a proximity criterion (see Section 3.4.1), an orientation criterion (see Section 3.4.2) and a colorimetric criterion (see Section 3.4.3). ω_p , ω_n and ω_c are weighting terms, fixed by the user. In our experiments we used $\omega_p = 0.3$, $\omega_n = 0.45$ and $\omega_c = 0.25$. These criteria allow to match the voxels which correspond to the same part of the surface, identified by an orientation and a texture. In case of large motions, the color is the most invariant feature. The proximity should only be a discriminating characteristic when several voxels satisfy the other terms of the distance function.

We immerse the binary volume V^0 in the EDT grid of V^1 , so that the EDT value associated to each surface voxel of V^0 represents its distance to the next pose at time t_1 . This distance is used to automatically define a *search radius* which corresponds to the maximum amplitude of the motion. For each surface voxel v_i^n we look through the surface voxels of V^{n+1} contained in this neighborhood and we select the voxel v_j^{n+1} which corresponds to the smallest result of the equation 3.3. Figure 3.3 shows an example of voxel matching. The positions of voxels v_i^n and v_j^{n+1} define a 3D vector. This vector is added to a vector field at the v_i^n position. This vector field is represented by the same structure as the voxel grid. Each square could contain one or several vectors. The same operation is repeated, looking this time, for each v_j^{n+1} , for the matching surface voxel v_i^n . The resulting vectors are added to the vector field at v_i^n position. This backward pass allows us to find a part of the motion which could have been ignored by the forward matching process (see Figure 3.5, top). Thus, we ensure that each surface voxel in V^n and V^{n+1} is associated to at least one vector.

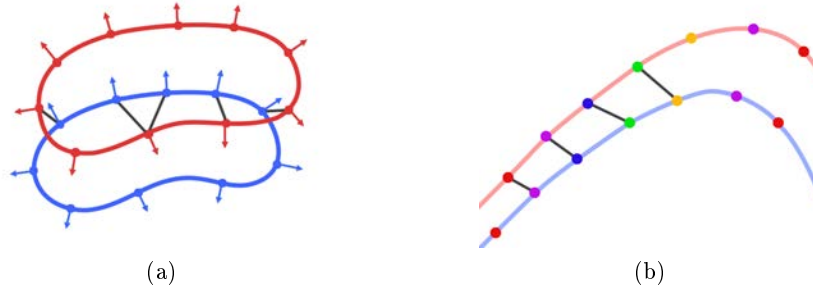


FIGURE 3.4 – Examples of inaccurate matching between two surfaces: (a) without normal criterion, the matching is not robust to large motions, (b) without colorimetric criterion, the matching is sensitive to tangential displacements.

3.4.1 Proximity criterion

The proximity criterion corresponds to the Euclidean distance between the two voxels:

$$\delta_{i,j} = \left\| \mathbf{p}_j^{n+1} - \mathbf{p}_i^n \right\|$$

with \mathbf{p}_i^n and \mathbf{p}_j^{n+1} being the 3D positions of v_i^n and v_j^{n+1} . This criterion allows us, if several voxels satisfy the other criteria, to select the closest one (see Figure 3.9b).

3.4.2 Orientation criterion

The orientation criterion represents the difference between the normal vectors of the two voxels:

$$\varphi_{i,j} = 1 - \mathbf{n}_i^n \cdot \mathbf{n}_j^{n+1}$$

with \mathbf{n}_i^n and \mathbf{n}_j^{n+1} being respectively the normal vectors at v_i^n and v_j^{n+1} . As illustrated in Figures 3.4a 3.9c, this criterion penalizes the matching of two voxels which belong to back facing surfaces. For example, in figure 3.3, the voxel (1) is matched with voxel (2) which normal vector has a closer orientation.

3.4.3 Colorimetric criterion

The colorimetric criterion is similar to a *block matching* algorithm, as used for motion estimation in digital video processing. We compare the colorimetric difference between two voxels as well as between their direct neighborhoods:

$$\sigma_{i,j} = \left\| v_j^{n+1} - v_i^n \right\|_{RGB} + \left\| B_j^{n+1} - B_i^n \right\|_{RGB}$$

B_i^n and B_j^{n+1} are the blocks which correspond to the surface voxels contained in a neighborhood of fixed size b around v_i^n and v_j^{n+1} , respectively:

$$B_i^n = \sum_{k=1}^b v_{i+k}^n$$

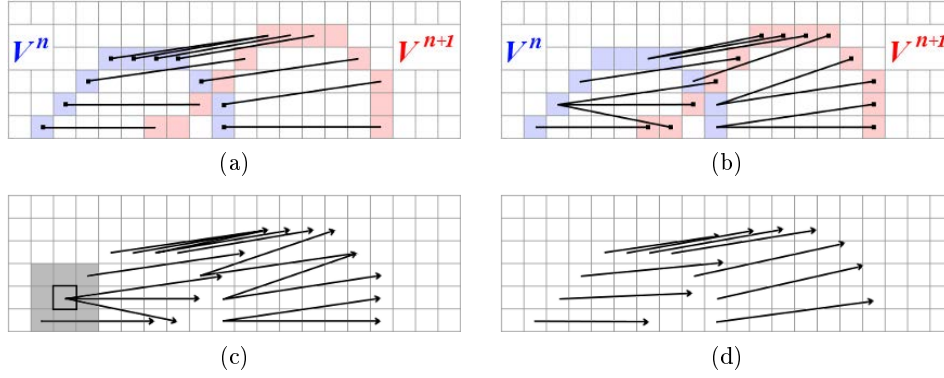


FIGURE 3.5 – Top: forward (a) and backward (b) matching between the two volumes. Bottom: Gaussian filter (in grey) applied to the raw vector fields (c) and final motion field (d).

if v_{i+k}^n belongs to the surface. This constraint favors the matching of two voxels which belong to close color blocks corresponding to the same object's part (see Figures 3.4b and 3.9d).

3.5 Motion flow regularization

The voxel matching step, explained in Section 3.4, results in a 3D vector field which roughly describes the motion of the volumetric object between V^n and V^{n+1} . We name this initial vector field $D1$. However, several inconsistent matches remain and the global motion is too irregular to be used. That is why a *smoothing* step is performed to get a coherent motion flow, as shown in Figure 3.5 (bottom). We apply a Gaussian filter on the initial vector field. For each surface voxel, we compute a single vector which is an average, weighted by Gaussian coefficients, of all the vectors in a defined neighborhood. This new vector field is named $D2$. Each vector $D2_i \in D2$ is computed by the following equation:

$$D2_i = \sum_{j=0}^M G D1_j \quad (3.4)$$

M being the neighborhood around $D1_i$. G is a Gaussian function:

$$G = \exp\left(\frac{-(x_i - x_j)^2 - (y_i - y_j)^2 - (z_i - z_j)^2}{\sigma^2}\right) \quad (3.5)$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) are the 3D coordinates in the grid of voxels v_i and v_j , respectively (*i.e.*, the squares which contains vectors $D1_i$ and $D1_j$). Each vector $D2_i$ is thus obtained by computing a weighted average of the vectors in $D1$ in the neighborhood M (a fixed size window centered on the voxel $D1_i$). This convolution is therefore applied as a discrete linear filter on the 3D grid. The value of σ is automatically deduced from the size of the kernel window, which is fixed by the user, as described in appendix C.1.

After this convolution, we obtain a smooth 3D motion field where each surface voxel is associated with a single motion vector. This filtering operation regularizes the vector

set to produce a coherent motion description where each surface voxel is associated to a single motion vector. The size of this filter (the window M) depends on the dimension of the volumes and must be defined by the user. The filter can also be applied several times to enhance the smoothing effect. In our case, we usually apply from 3 to 7 iterations, depending on the dataset (see Section 3.7.1).

3.6 Implementation and improvements

The previous sections described the brute force algorithm. We now present several enhancements which improve the quality and regularity of the initial matching (see Section 3.6.1) and the computing time (see Section 3.6.2).

3.6.1 Limitation of multiple matchings

In order to avoid as much as possible the multiple matching, the initial matching proceeds in two steps. In the first step, for each voxel $v_i^n \in V^n$, we look for the best matching voxel $v_j^{n+1} \in V^{n+1}$. If v_j^{n+1} is already matched with another voxel v_k^n from V^n , we compare the distances (v_i^n, v_j^{n+1}) and (v_k^n, v_j^{n+1}) (corresponding to the equation 3.3). If the distance (v_i^n, v_j^{n+1}) is the smallest, these two voxels are matched together and the voxel v_k^n is unmatched. In the other case, v_k^n stays matched with v_j^{n+1} .

At the end of this first stage, we obtain a partial matching between the surface voxel of V^n and V^{n+1} without any multiple association. However, some surface voxels remain unmatched. In the second step, these unmatched voxels must be linked to a surface voxel of the other frame. For each unmatched voxel v_i^n , we go through its neighborhood $N(i)$. We note v_l^n the surface voxels of V^n which belongs to $N(i)$. For each voxel v_l^n which is matched with a voxel $v_m^{n+1} \in V^{n+1}$, we compute the distance (v_i^n, v_m^{n+1}) . The unmatched voxel v_i^n is then matched with the voxel v_m^{n+1} which minimizes this distance.

3.6.2 Prediction

Considering that, at a time t_i , the motion flow between t_i and $t_i + 1$ should be a continuity of the previous motion vectors (between t_{i-1} and t_i), we improve our voxel matching algorithm once the motion vectors between the first two frames of the sequence has been estimated. Each surface voxel at t_i uses the motion vector previously estimated between t_{i-1} and t_i to predict the position of the matching voxel in t_{i+1} . We then search for the best matching voxel in t_{i+1} using a noticeably reduced radius around the predicted position, offering a highly efficient speedup of 60% compared to the brute force algorithm (see Section 3.7). This prediction is repeated for the next matching phases throughout the whole sequence.

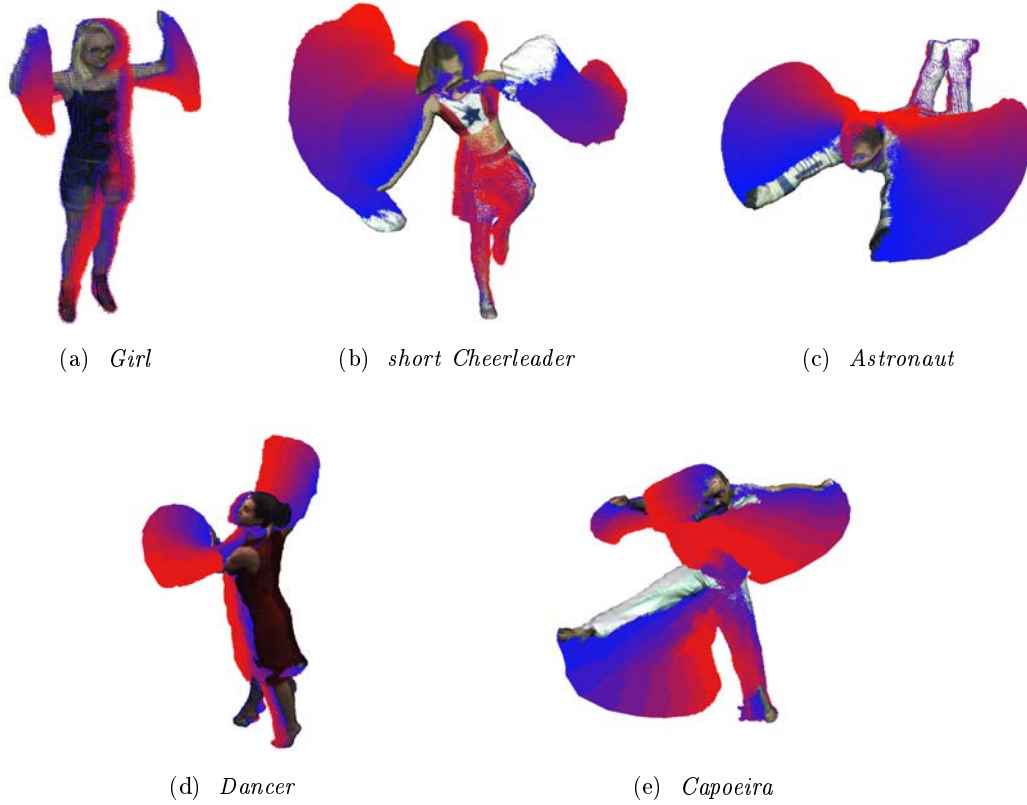


FIGURE 3.6 – Accumulated motion flows through several frames of five test sequences (motion vectors are oriented from blue to red).

3.7 Results

We evaluate our method on several datasets obtained through volumetric visual hull reconstruction. The *girl* dataset contains simple motions, with a woman slowly moving her arms. The visual hull volume has a $73 \times 132 \times 43$ voxels resolution and is reconstructed for 30 frames. The first mesh, used as a template, contains 11912 vertices. We also applied our algorithms on several reconstructions of actors' performances described in Chapter 5 (see Section 5.2). The *short cheerleader* and *astronaut* sequences both contain 25 volumes, with an average $180 \times 270 \times 170$ voxel resolution. The *astronaut* contains a nearly-rigid but large motion of the arms. The *cheerleader* is more challenging due to the free-moving shapes of the pom-pom, the skirt, and the large motions of the arms. These two datasets were generated by the RECOVER 3D studio. We also used the *dancer* and *capoeira* sequences, reconstructed from multi-view videos acquired by other infrastructures (*GrImage*¹⁵ and *MPI Informatik*¹⁶, respectively. See Section 5.2 for details). The *dancer* dataset represents a quick dancing motion, with large motions of the arms and a dress, with a lower quality of the visual hull. The *capoeira* sequence contains fast motions, especially for the legs, which lead to large inter-frame displacements. The quality of these visual hulls also suffers from the low resolution, number of viewpoints, and the inconsistencies of silhouettes' masks. All timings were measured on a 64 bit Intel Core i7 CPU 2.20 GHz.

15. <http://4drepository.inrialpes.fr/>

16. <http://resources.mpi-inf.mpg.de/siggraph08/perfcap/>

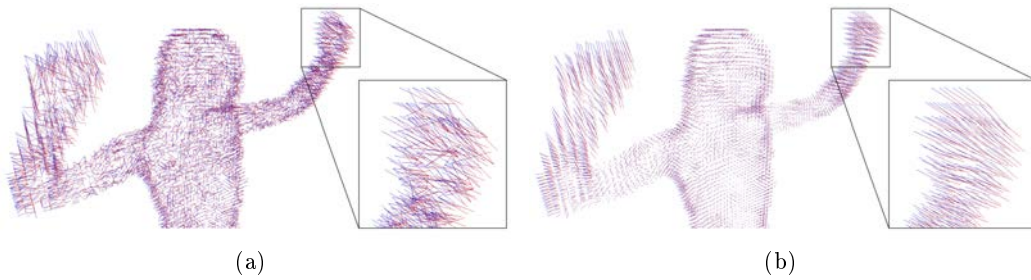


FIGURE 3.7 – Motion field regularization. (a) result of the voxel matching step. (b) vector field after regularization (vectors are oriented from blue to red).

Sequence	Cheerleader (short)	Astronaut	Dancer	Capoeira	Girl
Voxel-matching step	19s	42s	40s	230s	7s
Regularization step	18s	17s	9s	18s	3s

TABLE 3.1 – Average motion flow computing times between two consecutive poses.

3.7.1 Evaluation of the motion flow reconstruction

When evaluating the motion flow on these datasets, we obtain a satisfying motion field due to the regularization step, where each surface voxel is associated to a displacement vector (see Figure 3.7). Figure 3.6 presents the results obtained after computing the inter-frame motion flows throughout the complete sequences. The computing times are presented in Table 3.1. One can observe the noticeably longer time required for the voxel-matching step with the *capoeira* sequences, due to the fast inter-frame motion. As described in Section 3.6.2, using the previous motion flow as a prediction to reduce the search radius allows to substantially decrease the computing time. For instance, the voxel matching between the first two poses (without prediction available) of the *short cheerleader* is computed in 51s whereas it is reduced to 17s between poses 2 and 3. We then apply the regularization step (see Section 3.5) with a 7^3 window and 5 iterations. The computing time for this second step, presented in the last row of Table 3.1, depends on the parameters of the filtering and on the volumes’ resolution (not on the motion’s amplitude). As we applied similar parameters, the timings are almost similar for the sequences which have comparable voxel resolutions.

We compared our approach with our own implementations of two 3D-adapted optical flow algorithms as presented in [19]: the first one is based on the Lucas and Kanade method [101] and the second one on the variational approach by Horn and Schunck [65]. For comparison, the method described in [14] is using a similar approach to the Lucas-Kanade version, applied on a discrete distance function, as our EDT (see appendix C.3). Our experiments show that for similar settings, the Lucas-Kanade approach is faster (less than 5 seconds for *girl*) but displacement vectors are not oriented correctly (see an example of results in Figure 3.8a for a close-up on the girl’s upper body). It was expected as this kind of image warping approach is not well suited to large displacements. One common improvement to avoid this problem would be to implement a coarse-to-fine computation. The Horn-Schunck algorithm is significantly slower (5 minutes on the same dataset) and does not give convincing results with displacement distances not corresponding to the actual movement (see Figure 3.8b). With the other datasets, these limitations of our Horn-Schunck

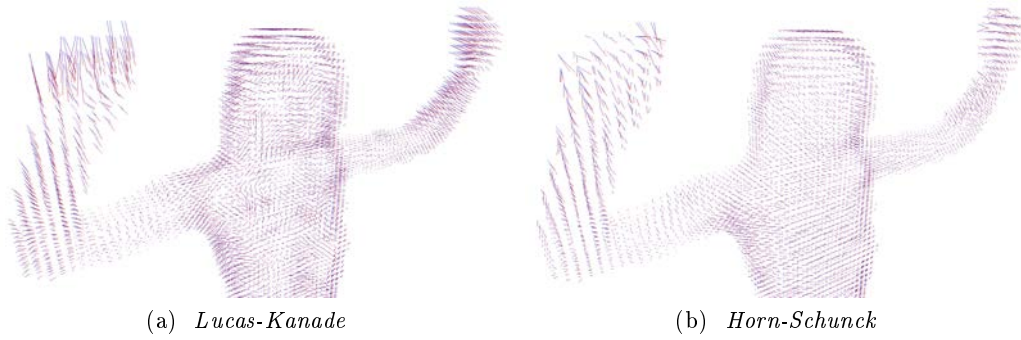


FIGURE 3.8 – Result of the two 3D optical flow computation between two frames of the *girl* sequence.

implementation are increased, due to the higher resolution of the volumes. We thus compared our motion flow computing with the Lucas-Kanade flows. Results are presented in the Table 3.2 for three datasets. With the other sequences, the motion's amplitude between two consecutive frames is too large to get a consistent flow with the Lucas-Kanade implementation. Instead, our back and forth voxel matching ensures that the motion flow covers the whole displacements, independently from their amplitude. The Euclidean distance volume, used as a 3D picture, does not seem to provide a sufficient information to compute a consistent motion information. Despite its high algorithmic complexity, our voxel matching method provides a better representation of the motion. While it is mostly only possible to evaluate visually the motion flows, a quantitative evaluation was performed on the mesh itself (see Section 3.7.3) which confirms our observations on the flows. Finally, compared to the 3D optical approaches, our method gives more consistent motion vectors by considering all the color information and orientation, instead of just the EDT. The computing time is satisfying (between the two other approaches). We also note that our voxel-matching step is only performed on surface voxels, while the 3D optical flows are computed on the complete EDT grid, which makes our method more appropriate to the data and more effective for higher resolutions.

3.7.2 Discussion on the chosen parameters

The method introduced by Nobuhara and Matsuyama [117], which uses only Euclidean distance (meaning, in our case, $\omega_n = \omega_c = 0$) is less efficient than the results we obtain with our multiple criteria approach. Figure 3.9 shows the influence of the three criteria (proximity, orientation, color) for voxel matching, defined by weights ω_p , ω_n , and ω_c (see Eq.(3.3)), defined by the user. Figure 3.9b shows that without the proximity criterion ($\omega_p = 0$), most of the matched voxels are too distant. The matching could associate two voxels which seem identical but do not correspond to the same part of the surface. The same problem appears if the orientation criterion's weight (ω_n) is set to zero. As illustrated in Figure 3.9c, most of the voxels are matched with another voxel which is close but corresponds to a backfacing surface. Figure 3.9d shows the lack of precision in the matching computed without colorimetric criterion ($\omega_c = 0$). The efficiency of this criterion increases when the volume is highly textured (*i.e.*, there are lots of variations in the voxels' colors). At last, Figure 3.9e shows that these criteria do not have the same influence, depending on the dataset used, and most of the time, different weights are chosen by datasets. These

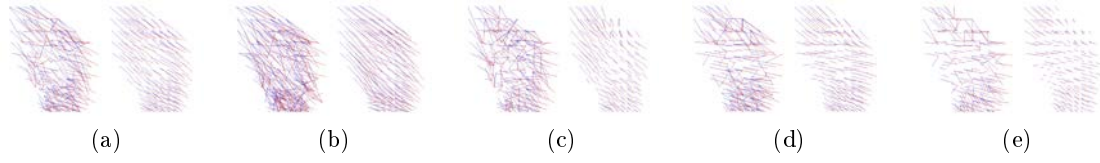


FIGURE 3.9 – Influence of the matching criteria. Results, before and after regularization, of the left hand’s voxel matching: (a) with $\omega_p = 0.3$, $\omega_n = 0.45$ and $\omega_c = 0.25$, (b) without proximity criterion ($\omega_p = 0$), (c) without orientation criterion ($\omega_n = 0$), (d) without colorimetric criterion ($\omega_c = 0$), and (e) with all weights set to 1.

results show that this voxel-based approach can be made drastically more robust for visual hulls if one considers orientation and texture of the voxels for matching and proper filtering. It is really the combination of the three criteria that improves the quality of the matching process. Note that in case of highly textured data (varied color patterns on the surface), the colorimetric criterion helps to stick to a proper voxel matching. Therefore, the weight ω_c should be increased. On the contrary, in case of poorly textured volumes (uniform colors), the orientation criterion should be the main hint for the matching. In any case, the proximity criterion should have the lower weight as it is only a discriminating term in front of the other criteria. The size of the Gaussian filter and the number of iterations could be increased for the processing of volume sequences with a higher voxel resolution. Note that increasing the size of the window or applying several iterations leads to a equivalent smoothing effect. We thus keep the filter’s size fixed with a constant value (between 5 and 9) and the user should only vary the number of iterations according to the data (we never used more than 10 iterations in our tests). At last, note that the successive iterations of the Gaussian regularization could also leads to an unnecessary over-smotting effect (we show in Chapter 4 that the final motion flow can be considered as a rough description of the displacements).

3.7.3 Mesh animation

After the motion flow is filtered, we use it to match a chosen template mesh (one of the sequence frames) to the sub-sequent meshes by pairs of frames, regularized using a mass-spring system in an iterative approach, in order to create a unique mesh, animated over time. The mesh animation is computed by a simple advection of the vertices by the motion flow (each vertex is moved according to the closest motion vector). A mesh regularization algorithm is eventually applied in order to maintain a consistent surface geometry. The template generation and the mesh regularization are described in Sections 4.4.1 and 4.5.1.2, respectively.

To measure the matching quality of the deformed template and the target pose, we used as metric the Hausdorff distance, which represents the distance between the deformed template and a mesh obtained by visual hull reconstruction of the same frame (this value is computed with respect to the diagonal of the bounding box). We used the MeshLab¹⁷ implementation, based on several distance computations between sampled surfaces and returning an average value. We evaluated the whole process with motion vectors obtained by our method (voxel matching) and by 3D optical flow with the same mesh regularization

17. <http://meshlab.sourceforge.net/>

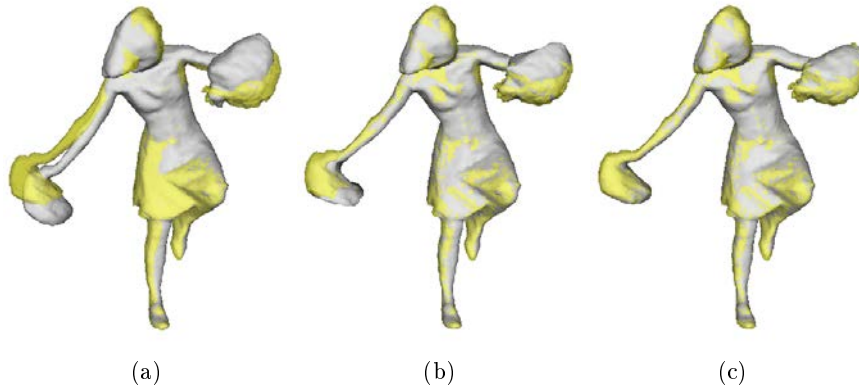


FIGURE 3.10 – Overview of the mesh animation process. Template mesh (in grey) in initial pose (a). The same mesh after the application of the motion vectors towards the next visual hull in yellow (b). The final result after mesh regularization (c).

parameters. The Table 3.2 presents the evolution of this matching metric during three sequences. We see that after several frames, the Hausdorff distance increases for the Lucas-Kanade approach whereas it stays stable with our method. These divergences correspond to the inconsistencies that appear in the mesh after several frames, due to an inaccurate motion flow. If the Lucas-Kanade approach and ours give similar results for the first poses, the results differ significantly from the real visual hull after several frames. With Lucas-Kanade, the mesh animation stays robust for 23 frames of the *girl* sequence. With our voxel matching approach, we obtain consistent results during the complete sequence. With the *dancer* dataset, the shape matching, using Lucas-Kanade, starts to produce inconsistent results after only 3 frames. Similar results are obtained with the *short cheerleader* where the mesh animation becomes inconsistent after 5 frames (see Figure 3.13b). Whereas with our motion flow, the Hausdorff distance stays stable (see Figure 3.11). The *girl* dataset is the only one where the mesh matching stays robust during the whole sequence using the two 3D optical flow approaches, even if the visual mesh consistency is quickly lost, because of the low motion's amplitude. With the other sequences, the 3D Lucas-Kanade implementation fails to maintain a consistent mesh animation after a few frames. The consistency of the matching between the animated mesh and the visual hulls is also measured by comparing the color of the voxels which correspond to each vertex during the sequence. We compute the distance in the colorimetric space between the closest voxels of a vertex in two adjacent frames and repeat this for each pair of frames through the whole sequence. The resulting values are normalized with respect to the maximum color difference (between black and white). The average difference is 0.042 for the *dancer* dataset (with a standard deviation of 0.059) and 0.039 for the *short cheerleader* sequence (with a standard deviation of 0.058). These values show that, despite the regularization of the mesh, the moving vertices stay globally associated with the same parts of the surface during the animation. The maximum differences are logically observed in the regions animated by the largest motions, as shown in Figure 3.12.

Frame	Girl		Dancer		Cheerleader (short)	
	Lucas-Kanade	Voxel Matching	Lucas-Kanade	Voxel Matching	Lucas-Kanade	Voxel Matching
2	0.00157	0.00157	0.007325	0.003321	0.002224	0.001772
3	0.00157	0.00157	0.008143	0.003091	0.002180	0.001846
4	0.00157	0.00157	0.007863	0.003061	0.002558	0.001914
5	0.00160	0.00160	0.006692	0.002173	0.002914	0.002244
6	0.00167	0.00167	0.006649	0.002682	0.002564	0.002196
7	0.00172	0.00173	0.006525	0.002569	0.002709	0.002054
8	0.00171	0.00173	0.007404	0.002431	0.002872	0.002150
...
12	0.00175	0.00177		0.003065	0.003166	0.001854
...
16	0.00180	0.00183		0.007644	0.003232	0.002061
...
20	0.00171	0.00174		0.005482	0.003198	0.002085
...
24	0.00177	0.00175		0.003628	0.002930	0.001945
...		
28	0.00206	0.00191		0.002933		

TABLE 3.2 – Mesh matching measurement (average Hausdorff distance) between animated mesh and visual hull of the target pose.

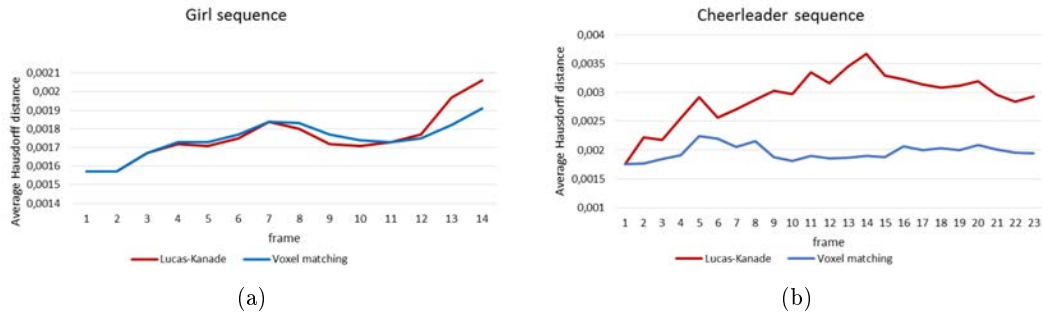


FIGURE 3.11 – Evolution of the average Hausdorff distance during the *girl* and *short cheerleader* sequences.

3.7.4 Limitations

The algorithms described in this chapter allow to compute a good estimation of the motion that occur between consecutive poses. However, these vectors rely on a set of correspondences that cannot ensure a perfect matching of the two shapes. Indeed, the motion flow gives a global information on the displacements but may locally include inaccurate vectors due to, for instance, outlier matches or inconsistent results caused by the regularization. Due to these limitations, the basic mesh animation described in Section 3.7.3 may leads to inconsistent displacements of the vertices. The other main problem of this system is that these vertices are moved independently, without assumption about the global motion of the surface. This often leads to inconsistencies on the mesh with the divergence or collapsing of the vertices during the animation. To handle these problems, an appropriate mesh animation procedure should both be able to follow the free-form motions of the motion flow and maintain a consistent surface. Our proposal for such a system is described

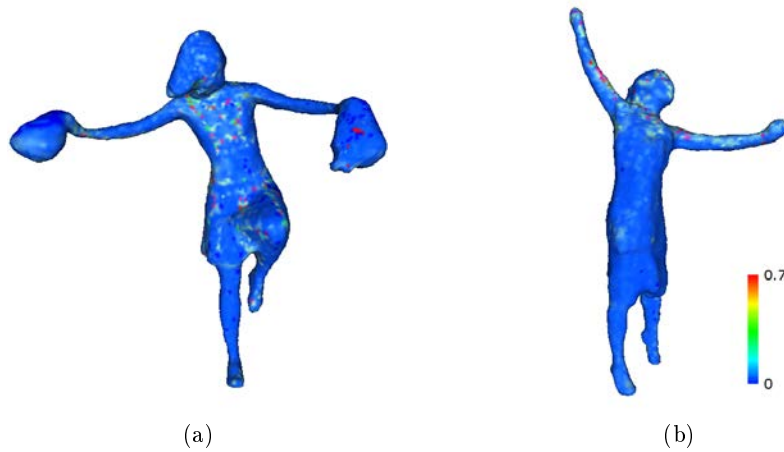


FIGURE 3.12 – Representation of the texture matching metric on the animated mesh for the *short cheerleader* (a) and *dancer* (b) sequences. Each vertex is associated with a color which represents the colorimetric difference of the closest voxels in two consecutive frames. A good matching corresponds to a small value. The minimum value is 0 (blue) and maximum is 0.7 (red).

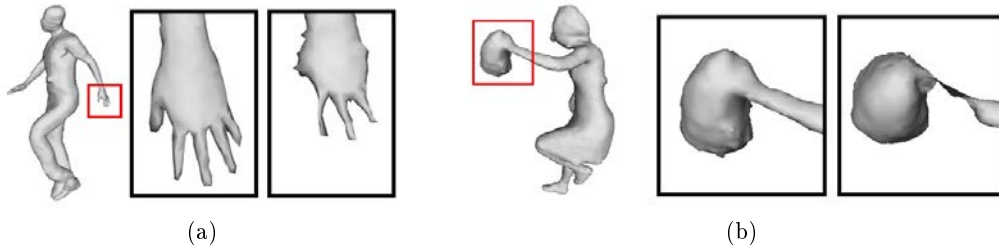


FIGURE 3.13 – **Stress case.** (a) Example of small details where the mesh slowly degrades over time by lack of dense enough information. (b) Other type of mesh drift occurs when the motion flow does not correctly match the new pose of the actor. Here is an example with the *short cheerleader* dataset, animated with the 3D Lucas-Kanade motion flow.

in the Chapter 4. Figure 3.13a shows an example on a synthetic dataset which contains thin details such as fingers. As described in Section 3.7.3, the animated mesh may become inconsistent if the motion flow does not properly match the successive actor's poses. These problems often occurs when we use the 3D Lucas-Kanade approach (as presented in Figure 3.13b) instead of our voxel matching algorithm, but can also occur after several frames even using our method, especially due to large displacements between two frames. A mesh animation based on an *ARAP* deformation [138], guided by the motion vectors, could ensure a better conservation of the mesh structure during the sequence and lead to more robust results. The datasets on which we evaluated our method are around 30-frame long. We wish to work on longer sequences in our future improvements. Another issue is the number of parameters which have to be defined by the user or empirically determined (weighting coefficients for voxel matching and mesh regularization, Gaussian filter radius, and number of iterations) and that may not be robust for all the sequence. These problems prevent us from computing efficiently an animation from long and complex sequences.

3.8 Conclusion

The proposed method allows us to compute a voxel matching for motion flow estimation. This correspondence is established without *a priori* knowledge about the nature of the volumes, except that they are of course supposed to represent the same object and belong to the same sequence. Our method first establishes an inter-frame correspondence, associating the voxels of two adjacent volumes from a time series of reconstructed poses. The correspondences are computed according to a matching function that relies on several criterions: the Euclidean distance, the texture color and the normal vector's orientation. These different measures ensure to associate voxels which belong to a similar point of the surfaces to match. The initial vectors field is then regularized by applying a Gaussian filter on it. The final result is a regular motion flow which describes the displacements of the reconstructed character between two consecutive poses. By applying this method between each pair of adjacent frames throughout the complete sequence, we obtain a 3D description of the actor's motion during the whole captured performance. Note that the motion flow may be used, as a descriptor of the actor's movements, in other applications such as interaction of the reconstructed character with its virtual environment. Compared to other methods, we demonstrated that our approach was generally more efficient and more robust when considering complex motion and long sequences.

Chapter

4

Mesh animation

Contents

4.1	Introduction	77
4.2	Positioning in the context of previous work	77
4.2.1	3D non-rigid registration	79
4.3	Pseudo-rigid deformation	80
4.3.1	Concept	80
4.3.2	Differential coordinates	80
4.3.3	As-Rigid-As-Possible mesh processing	81
4.4	Animation process	82
4.4.1	Template mesh	82
4.4.2	Anchors' selection	82
4.4.3	Pseudo-rigid mesh animation	84
4.5	Local optimization	89
4.5.1	First approach: numerical integration	89
4.5.2	Second approach: energy minimization	90
4.6	Results	92
4.7	Conclusion	93

Résumé

Dans ce chapitre, nous décrivons une méthode d'animation guidée par le champ de déplacements obtenu précédemment dans le Chapitre 3. Notre objectif est d'appliquer des déformations, guidées par les vecteurs de mouvements, sur un maillage initialisé à la première pose de l'acteur afin de l'aligner avec les poses successives de la séquence d'enveloppes visuelles. Nous utilisons comme maillage de référence la pose de l'acteur reconstruite à partir du premier volume de la séquence, à l'état t_0 (cf. Section 4.4.1). Nous partons du principe que la position initiale de l'acteur exclut toute ambiguïté topologique afin de nous assurer une surface de référence adaptée à la morphologie du personnage. La surface de l'acteur est triangulée grâce à un algorithme de *marching cubes* appliqué sur le volume puis régularisée pour obtenir un *template*. Au cours de l'étape d'animation, le maillage initial est plongé dans le champ de mouvements pour appliquer les déplacements. Nous utilisons la méthode de déformation de maillage dite ARAP décrite par Sorkine et Alexa [138] car elle permet de déformer un maillage en restant fidèle à la forme initiale : la surface est déformée tout en minimisant de manière globale la différence entre le maillage de la pose de référence et la nouvelle position du personnage.

On sélectionne à la pose t_i un ensemble de sommets qui doivent servir de points d'ancrages. On extrait à cette fin les sommets qui sont associés aux vecteurs de mouvements de plus forte amplitude. Cette sélection est effectuée sur un ensemble réduit de sommets préalablement échantillonnés de manière uniforme sur la surface (*Poisson disk sampling*). On récupère ensuite le vecteur de mouvement correspondant à chacun de ces sommets, ainsi que la valeur du *matching* qui lui est associée (cf. Chapitre 3). Cette valeur est utilisée comme score pour définir le poids de ce point d'ancrage lors de l'étape de déformation globale suivante. Etant donné que le flot de mouvements peut être fortement bruité, nous utilisons une méthode variationnelle en cherchant à déformer un maillage M' par l'application de transformations localement rigides, tout en conservant autant que possible la position des points clés (cf. Figure 4.6).

Nous cherchons à minimiser l'expression suivante : $E(M') = E_{ARAP}(M') + E_{ANC}(M')$ où E_{ARAP} est l'énergie de la déformation *As-Rigid-As-Possible* (cf. Section 4.4.3). La minimisation de ce terme permet de conserver les coordonnées différentielles des sommets du maillage au cours de l'animation. On s'assure ainsi que la déformation de la surface n'altère pas la structure des triangles (contrainte de cohérence temporelle). E_{ANC} est une énergie quadratique appliquée sur les points d'ancrages qui permet d'adapter l'influence des points d'ancrage sur la déformation globale en fonction de leur poids. A partir des conditions d'optimalité des énergies ci-dessus, on déduit des expressions précédentes un système linéaire. La résolution du système (cf. Equation 4.4) s'effectue alors de manière itérative (cf. Section 4.4.3.2). L'étape précédente nous a permis de déformer le maillage initial pour lui faire adopter la pose suivante. Cependant, après cette transformation globale, une étape d'optimisation locale est encore nécessaire pour que le maillage soit parfaitement adapté à la silhouette de la nouvelle pose (cf. Section 4.5.2). En effet certaines sections de la surface de l'objet sont animées de mouvements libres qui n'ont pas été détectés lors des étapes précédentes. Pour aligner le maillage avec l'enveloppe visuelle et pour supprimer les irrégularités qui peuvent survenir lors de la déformation, nous appliquons un algorithme basée sur une minimisation d'énergie comparable à celle de la déformation globale : une énergie E_{DATA} entraine chaque sommet vers la surface de l'enveloppe visuelle tandis que l'énergie E_{ARAP} assure toujours la contrainte de rigidité locale.

4.1 Introduction

In this chapter, we describe our proposed solution for the mesh animation process. Using the motion flow which describes the movements of the actors (see Chapter 3), the next step is to animate a template mesh according to these displacements, leading to a dynamic mesh. Our goal is to develop a mesh animation method which deforms a template mesh from one pose to the successive one, fitting the input data given by the input visual hulls (see Chapter 3, Figure 3.1). Our surface deformation should also handle free-form motions while maintaining a consistent mesh structure throughout the animation. We first give a brief overview of the mesh animation methods that can be used in the context of multi-view reconstruction (see Section 4.2). We then describe a pseudo-rigid deformation method (see Section 4.3), based on as-rigid-as-possible processing, driven by a reduced set of anchors which lead the displacement of the whole surface (see Section 4.4), following the hint of the motion flow. The last step is a local optimization that handle the inter-frame surface details evolution (see Section 4.5). A summary of the preliminary results is described in Section 4.6.

4.2 Positioning in the context of previous work

Several types of mesh animation approaches could be applied in the context of multi-view reconstruction. *Skeleton-based* animation techniques have been shown especially effective for motion capture. The skeleton is a set of articulated segments that drive the motions of a surface, by analogy with the bones that articulate the limbs of a body. Each segment is animated by rigid transformations (translations and rotations) and linked to one or several other *bones* at its extremities. Limited degrees of freedom define the articulation (*joint*) between two segments. The surface mesh is then associated with the skeleton. The motion of each vertex is interpolated from the motion of the closest bones. This operation is named *skinning*. The most basic way to perform it is to use a linear interpolation according to the distance between the vertex and each bone. This method is named *Linear Blend Skinning* (LBS) or *Skeletal Subspace Deformation* and has been first described by Lewis et al. [89]. Other skinning algorithms were proposed, such as *Multi-Weight Enveloping* (MWE) [169]. However, skeleton-based reconstruction cannot handle complex, non-rigid motions, such as free-form displacements.

For this purpose, *cage-based* animation is seemingly more appropriate [157]. A cage is a set of simple polygons (acting as exo-skeletons) in which the model is embedded. From the motion of the cage, the displacements of the mesh vertices are directly interpolated. Cage animation can be described as a kind of a *space deformation* technique where a polygon is manipulated with a reduced set of control point and deforms the space it contains. Each point inside the cage is expressed as the affine sum of the cage's vertices, using, for instance, *Mean Value Coordinates* (MVC) [72], *Harmonic coordinates* [71], or *Green coordinates* [95]. This leads to a shape-preserving deformation, less constrained than a skeleton-based animation. It is a good way to modify the global shape of an object, with no alteration of the surface details. However, it is unable to apply local deformations on the surface, unless building a sensibly more complex cage model.

Keyframes are sometimes used instead in, *e.g.*, facial animation [88] to represent key

poses of a model. Mesh vertices are interpolated as linear combinations of keyframes. This approach is only valid when the range of motion of a model is known and limited, making it impractical for our case in which actors can move arbitrarily. This method can also be seen as a *morphing* between two adjacent frames. Morphing is a blending operation that transforms a source shape into a target one. It can be processed through interpolation if the two shapes are close from each other and share the topology and number of vertices. The mapping of two shapes can also be processed with a parameterization of the two surfaces on a common domain. For topological spheres (genus 0 surfaces), the manifolds are naturally embedded into the unit sphere [3]. For more complex shapes, a segmentation of the surface into several pieces may be required, the different parts being then mapped separately and morphed locally. However, the morphing of 3D objects with different topology and connectivity is a challenging task which may involve a remeshing of the shapes.

Active contour models are another way to track non-rigid moving objects, using a deformable mesh to match a surface defined by an implicit function. Terzopoulos [151] was the first to describe geometric curves, constrained by physical forces, to fit a data in an image. In follow-up work by Kass *et al.* [74], the snakes are synthetized as curve models, for extracting a contour in 2D pictures. They are based on two energies: an internal energy for regularization and an external energy for data fitting. Next, an iterative algorithm updates the shape of the snake by a physical simulation. The points of the curve are pushed to the contour, according to the external energy, while the internal energy retains the shape's consistency. The many implementations of snakes can be separated in two approaches: *implicit representations* [118] and *explicit representation* (e.g., *deformable models* [152, 41]). In explicit approaches, the deformable model is initialized with a starting shape, modeled as an elastic surface which vertices are submitted to several forces. The vertices' position are then iteratively updated following these constraints, through a physical simulation, to match the target shape defined by the data. Several methods were proposed to develop snake models that can modify their topology during the deformation [48]. Particularly, δ -snake models [82, 22] update their connectivity to match the target surface with a triangulated mesh, eventually changing their topology. Snake models are limited by the locality of the data: the external energy is provided by the region of the data where are located the snake's points. The snake's position is then updated according to these data to start a new iteration. This way, the model may fall into local extrema, especially in the case of large distance between the initial position and the target shape. It can also be difficult to adjust the influence of each energy (regularisation *vs* data), especially in case of noisy data.

Kilner *et al.* [77] addresses these limitations with a *dual-mode deformable model*. This snake works with two modes: a *search mode* which seeks out a consistent set of data points to perform the reconstruction and a *fitting mode* with a usual snake algorithm (with a much weaker regularization force as most of the outliers have already been discarded by the previous mode). Such deformable models have already been used to merge silhouette-based reconstruction with stereo-matching constraints in several model-free reconstruction approaches [57]. However, in our case, the goal is to deform a given shape to the next frame's pose and in case of large motions, active surface are not suited to match a far target surface. Moreover, these dynamic models typically modify the mesh connectivity in time to match the target pose, which prevents us from keeping a consistent mesh throughout the whole animation.

Recently, *As-Rigid-As-Possible* (ARAP) surface modeling [138] offered an interesting alternative allowing to globally deform a mesh with fixed connectivity with local geometric

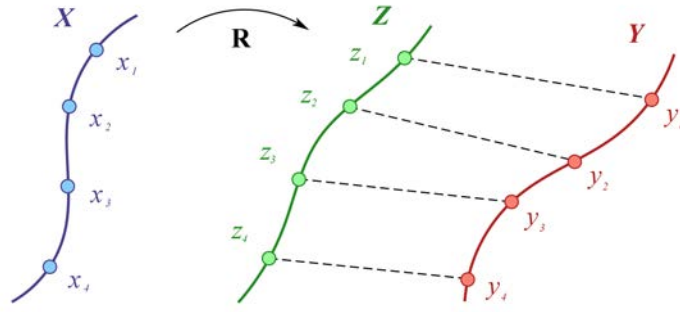


FIGURE 4.1 – 3D registration.

transformations that are as close to rigid as possible. A set of anchor points drive the global deformation of the mesh through a non-linear, but simple energy minimization. This approach is usually employed in a geometric modeling framework [123], but also presents suitable properties for animation [103] and we can adapt its use to our context.

4.2.1 3D non-rigid registration

The goal of a *pairwise registration* is to align a source model onto a target model. In our case, the model is a 3D discretized surface, *e.g.*, a triangulated mesh. The goal of the registration is to find a deformation that aligns the source with the target, driven by a set of correspondences computed between the two models. The source is denoted $X = \{x_0, \dots, x_n\}$ and the target $Y = \{y_0, \dots, y_n\}$. A new surface $Z = \{z_0, \dots, z_n\}$ is described as the deformed version of X . As the registration is most of the time an iterative process, the surface Z is initialized with the geometry of X and iteratively deformed until it matches the target Y . The registration of a surface can be generalized as the minimization of an energy:

$$E_R(Z) = E_{MATCH}(Z) + E_{PRIOR}(Z), \quad (4.1)$$

As described in [28], E_R is the sum of a matching energy E_{MATCH} that measures the displacement distance between the source and the target (*i.e.*, data fitting term) and a prior energy E_{PRIOR} that represents the geometrical properties that the surface must respect during the deformation. The most usual 3D registration algorithms compute a rigid transformation to align static models, like in the ICP algorithms for instance. With dynamic surfaces, a non-rigid alignment is necessary. In this case, the matching energy is defined by a set of correspondences computed between the source and the target which define a non-rigid alignment. The prior energy then defines the way the source surface can be deformed to match the target. To keep a consistent shape, this deformation behavior often respects a *local rigidity* constraint (opposed to the *global rigidity* of ICP). This prior can follow an articulated scheme, like in skeleton-based animation for instance. The Figure 4.1 describes the registration process. The initial surface X is deformed to match the target Y . The transformation R can follow a rigidity prior. In this case, a single (R, t) transformation (rotation + translation) is applied on the complete surface. In case of free-form deformation, a transformation R_i is associated to each of the n points x_i , $i \in [0, n]$, of the initial surface. The transformation is driven by a set of correspondences between the transformed shape of X , noted Z , and Y .

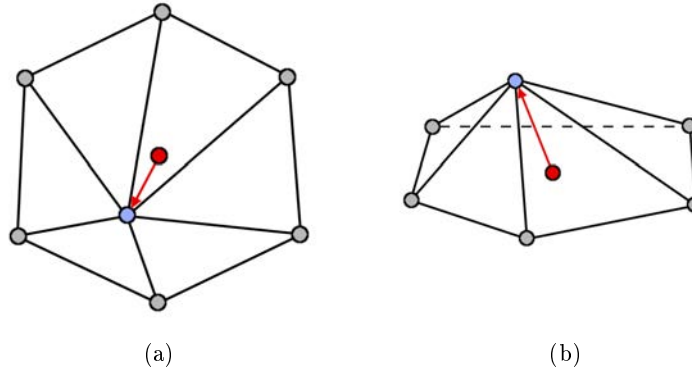


FIGURE 4.2 – Laplacian vector on a triangle mesh.

4.3 Pseudo-rigid deformation

We now present our proposed non-articulated 3D registration (see Section 4.3.1) applied for inter-frame mesh deformation. We also describe the differential coordinates (see Section 4.3.2) and how it can be used as local rigidity prior (see Section 4.3.3).

4.3.1 Concept

Articulated prior cannot handle free-form motions. This kind of deformation can be managed with *per-vertex* animation. However, in this case, a deformation prior has to ensure that the structure of the surface will not be damaged by the displacement of the vertices. A *local rigidity* is often desired. Among several types of linear variational surface deformation methods [27], Laplacian-based approaches present the interesting properties in the conservation of the local rigidity of the deformed mesh and their robustness to several type of displacements. In our case, the initial geometry X is a triangle mesh M containing n vertices v_i with $i \in [0, n]$. For this kind of discretized surface, a local rigidity behavior can be defined by the conservation of *differential coordinates*, as defined in the family of as-rigid-as-possible surface deformation approaches.

4.3.2 Differential coordinates

Considering a mesh M with a set of n vertices noted v_i as $M = \{v_0, \dots, v_n\}$. The *differential coordinates* of the vertex \mathbf{v}_i are computed with the difference between the absolute coordinates of \mathbf{v}_i and the center of mass (or barycenter) of its immediate neighborhood (see Figure 4.2):

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j)$$

with $N(i)$ the direct neighborhood of \mathbf{v}_i and d_i the valence of \mathbf{v}_i (*i.e.*, the size of $N(i)$). These coordinates can be viewed as the discretization of the continuous Laplace-Beltrami operator applied on the mesh (*i.e.*, a discretized surface). The differential coordinates are invariant under translation but sensitive to linear transform (*e.g.*, rotations). It can

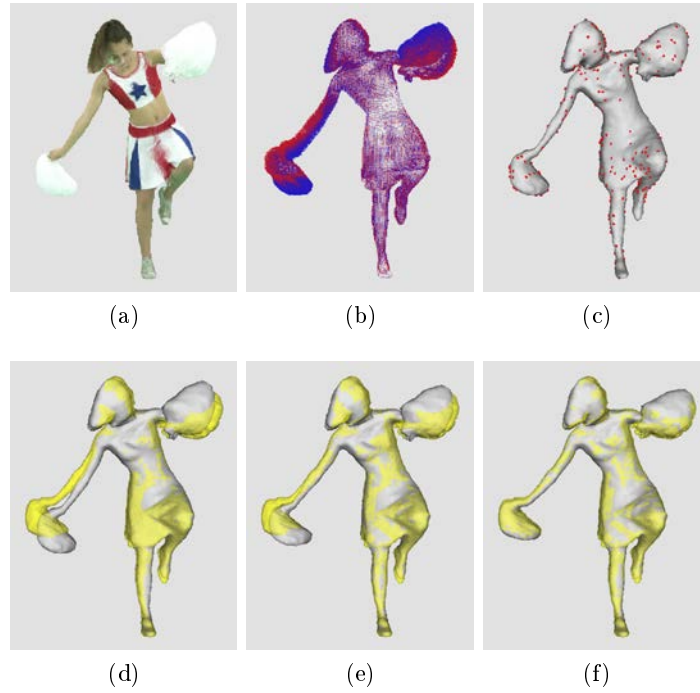


FIGURE 4.3 – **Inter-frame animation process.** Starting from two consecutive frames in a sequence of colored volumes (a), we first compute a motion flow (b). We then sample a set of anchor vertices (c) on the current mesh (in grey). To deform the current mesh toward the next visual hull in yellow (d), we perform a global deformation of the mesh based on the displacements of the anchors (e), before applying a local optimization on the mesh to finely match the target visual hull (f). This process is repeated through the whole sequence to match successive poses.

be represented as a 3D vector between the position of \mathbf{v}_i and the barycenter of its 1-neighbors, named *Laplacian vector*. The Laplacian vector represents the details of the surface locally. One notices that this vector may be used as a mesh smoothing operator (*i.e.*, high frequencies removal), often named *Umbrella operator* [78]. The *Laplacian matrix* L can be considered as the Laplacian operator applied on the mesh (*i.e.*, a discretized surface).

4.3.3 As-Rigid-As-Possible mesh processing

The ARAP algorithm defines a locally rigid deformation behavior based on the conservation of the differential coordinates of the vertices: in the ARAP deformation process, the registration is applied with the prior term focused on the conservation of the differential coordinates. The matching term relies on the set of anchors which drive the deformation. The term E_{PRIOR} from the equation 4.1 can thus be written in the following form:

$$E_{ARAP}(Z) = \sum_{i=1}^n \sum_{j \in N(i)} \|(z_i - z_j) - R_i(x_i - x_j)\|^2$$

with $x_i \in X$ and $z_i \in Z$. In the usual ARAP deformation applications, the anchors are manually moved and are thus considered as a strong constraint. In our case, these anchors

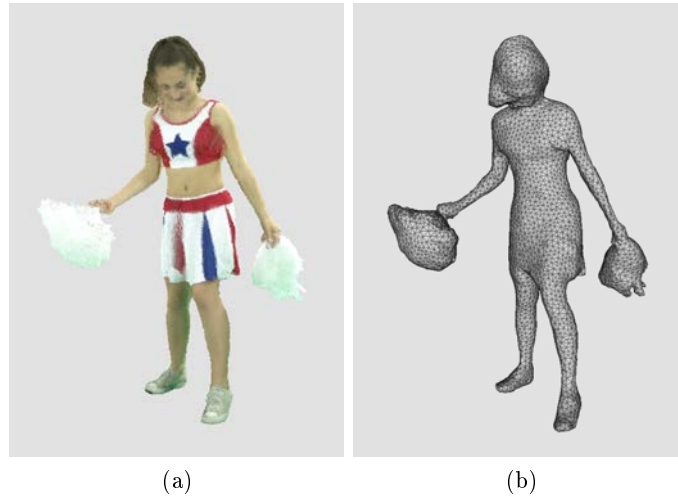


FIGURE 4.4 – First volume of the *long cheerleader* sequence (a) and associated template mesh (b).

must be automatically relocated.

4.4 Animation process

Our animation process, summarized in Figure 4.3, starts with the initialization of the template surface (see Section 4.4.1). The anchor vertices are then sampled on the mesh (see Section 4.4.2) and advected with motion vectors. The mesh deformation is driven by these anchors afterwards (see Section 4.4.3). These two last steps are repeated for each pair of pose throughout the complete sequence.

4.4.1 Template mesh

As described above, the 3D reconstruction used for our input data is a silhouette-based approach. Indeed, our mesh tracking algorithm has to successively match a sequence of volumetric visual hulls. We construct an initial (template) mesh based on the first volume of the sequence by extracting it using a *marching cubes* algorithm over the alpha value. A Laplacian smoothing [150] and a mesh simplification [94, 32] are then applied. The resulting triangle mesh may also be cleaned up through edge collapse simplification to ensure that each triangle is non-degenerate, and will thus not create numerical artifacts in our subsequent tracking (see Figure 4.4b). The objective is to use the surface extraction method already applied to compute the mesh sequences, as described in Chapter 1 (see Section 1.3.2).

4.4.2 Anchors' selection

We select a set of vertices at time t_i to be *anchor* points. These anchors are distributed over the surface. In an initial implementation, we selected the vertices to be anchors if

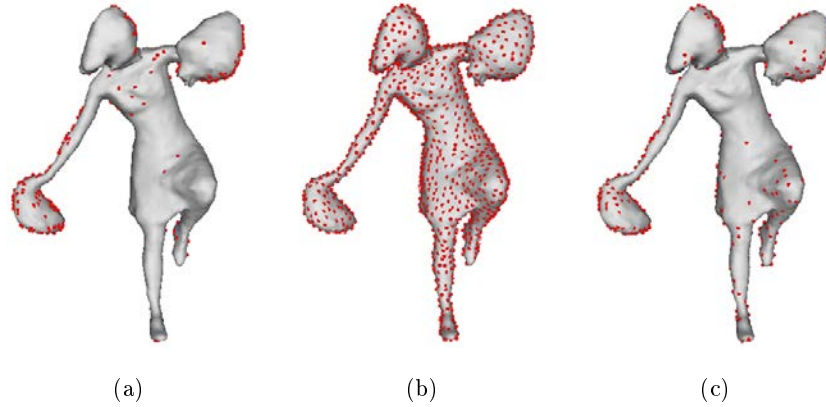


FIGURE 4.5 – **Anchor vertices selection:** vertices automatically selected using the EDT criterion only (a). Poisson disk sampling on the complete mesh (b). Anchors selection applied only on sampled subset of vertices (c).

they belong to a high curvature area of the surface. The surface's curvature is computed using the method described in [120] *via* spectral analysis of the covariance matrix of the one-ring neighborhood. Nevertheless, this strategy was not adapted to our data and often led to select points associated with artifacts and/or details of the surface which were not convincing as anchors.

Instead, as these anchors must drive the global deformation of the character, we select the vectors associated with the largest displacements. Note that a few anchors in static regions are also essential to guarantee that the immobile parts of the actor's body are not deformed. We use the t_{i+1} EDT grid to identify a set of vertices associated with the largest EDT_{i+1} values. The mesh's vertices at t_i are ordered according to their corresponding EDT_{i+1} values. Next, we select a fixed percentage of the highest ones. These vertices, being the most distant from the next pose, will naturally correspond to a large motion. The number of anchors is empirically fixed to 5% of the total number of vertices in the mesh (4% associated to maximal values and 1% of nonmoving vertices associated to minimal values). The mesh deformation could then be more or less constrained with respectively higher or lower number of anchors. In the same way, we also select a random subset of nonmoving vertices to avoid the over-deformation previously described. The correspondence score of the anchors, defined by the matching function described in Chapter 3, equation 3.3, will then be used as weights to adjust how strong we enforce the matching of these anchors in the global deformation step. Note that while this anchors' sampling method is particularly suited to our context, it could easily be adapted to other model-based approaches.

The result of this anchor vertices' sampling is shown in Figure 4.5a where anchors are figured by red dots. It can be noticed that these anchors are actually located on the mesh's sections which undergo displacements. However, our algorithm leads to a high concentration of anchors where the mesh is associated with important motions (here the hands and pom-poms). Instead we would like the anchors to be more widely distributed on the surface to cover all the motions occurring on the shape (*e.g.*, arms and skirt). We implemented a *Poisson disk* sampling algorithm [31] to obtain a subset of anchors regularly distributed on the surface. This algorithm allows to select a sample of vertices that cover the whole surface. Each sampled vertex is the center of a disk of fixed radius

(5 times of a voxel's size in the example presented in Figure 4.5) where no other vertex is sampled. This criterion ensure the regularity of the sampling. Our implementation uses the 3D grid structure of the volume to distribute the samples on the mesh and is described in appendix B.2. The result of this sampling is presented in Figure 4.5b. By applying our anchors' automatic selection on the subset of the vertices sampled with the Poisson disk algorithm, we finally obtain a set of anchors that correspond to the moving sections of the mesh while avoiding the anchors redundancy in the extremums of the motion flow (see Figure 4.5c). This process is detailed in Algorithm 1. Note that the Poisson sampling is processed only once, on the template mesh. The anchors' selection is then performed on this constant set of sampled vertices, at each pose of the sequence.

```

Data: mesh, EDT 3D grid
Result: list of anchor vertices
Create distanceList, vertSampList, anchorList;
nbVertices  $\leftarrow$  mesh.getNbVertices();
// get the result of vertices' sampling in the vertSampList
vertSampling  $\leftarrow$  PoissonSampling(mesh);
// fill distanceList with EDT values of sampled vertices
foreach element  $v$  of vertSampList do distanceList.add(EDT[ $v_x$ ][ $v_y$ ][ $v_z$ ] ) ;
// sort distanceList into ascending order
Sort(distanceList);
compteur  $\leftarrow$  0;
i  $\leftarrow$  0;
// get 3 percents of the vertices associated to the minimal values
while compteur < nbVertices*0.03 AND i < distanceList.size() do
    | anchorList.add(distanceList[i]);
    | compteur ++;
    | i ++;
end
compteur  $\leftarrow$  0;
i  $\leftarrow$  distanceList.size();
// get 2 percent of the vertices associated to the maximal values
while compteur < nbVertices*0.02 AND i > 0 do
    | anchorList.add(distanceList[i]);
    | compteur ++;
    | i ++;
end
return anchorList;

```

Algorithm 1: Anchors' selection

4.4.3 Pseudo-rigid mesh animation

The template mesh at t_i needs to be advected in the motion flow. We proceed in two main steps: a global deformation of the mesh derived from the sparse set of anchors (see Figure 4.6), followed by a fine adjustment of its vertices. The motion flow provides a global deformation field to be applied on the template but not a precise motion to apply on each vertex. We thus only consider the displacement of the reduced set of sampled anchors. In order to be resilient to noisy motion flows, we use a variational method to our anchor-

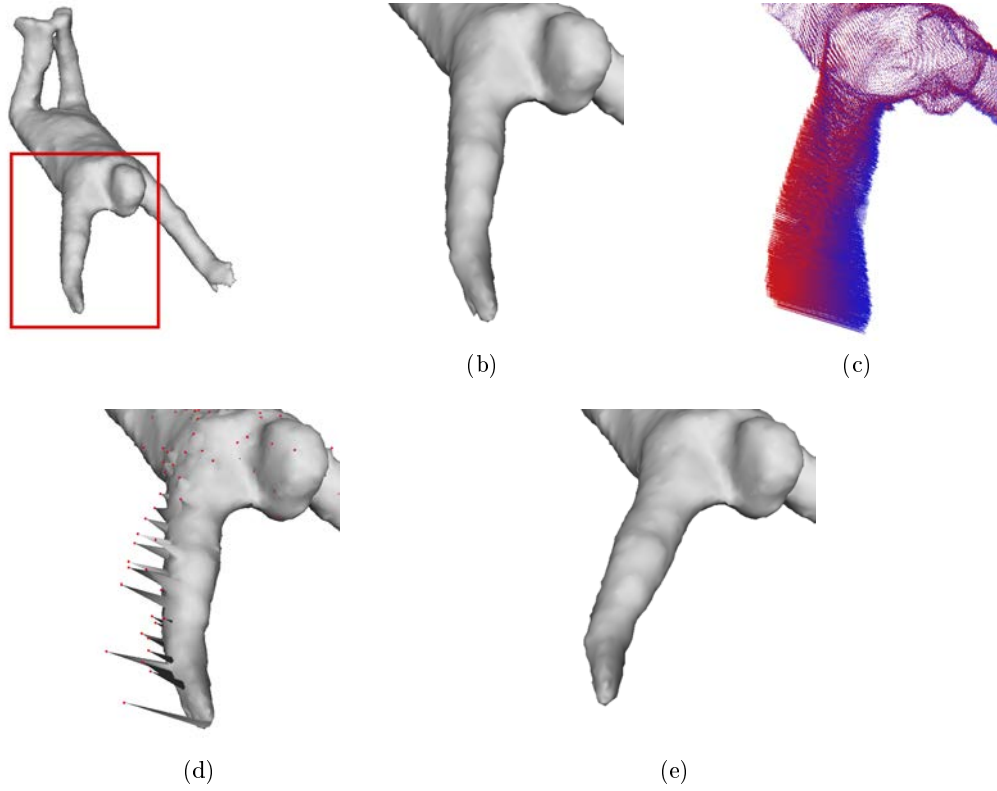


FIGURE 4.6 – **Pseudo-rigid mesh deformation.** Template mesh of the *astronaut* sequence at the initial pose t_i (b) and motion flow (vectors oriented from blue to red) computed between t_i and t_{i+1} (c). Motion applied to anchors vertices only (red dots) (d). New pose of the template after the anchor-driven as-rigid-as-possible deformation (e).

based global mesh deformation by searching for a deformed mesh M' with locally rigid transformations, while retaining as much as possible the final positions of anchor points (see Figure 4.6e).

4.4.3.1 Formulation

We minimize the following energy:

$$E(M') = E_{ARAP}(M') + E_{ANC}(M'),$$

where E_{ARAP} is the as-rigid-as-possible energy

$$E_{ARAP}(M') = \sum_{i=1}^n \sum_{j \in N(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2, \quad (4.2)$$

with $N(i)$ denoting the one-ring neighborhood of i . The terms p_i and p'_i represent the 3D positions of the vertex i , before and after applying the local transformation R_i . The

weight w_{ij} , associated with the edge between p_i and p_j , can be computed according to the cotangent weight method, or simply set to 1.

E_{ANC} is a quadratic energy measuring the error in the displacement of anchors

$$E_{ANC}(M') = \sum_{i=1}^n w_{a_i} \|p'_i - p_i\|^2, \quad (4.3)$$

where the weight w_{a_i} represents the degree of confidence given to an anchor point, described in Chapter 3 and given by the equation 3.3. The distance D_i given by equation 3.3 is a normalized value which is low for a good match and high for a bad one. Here, the weight of the anchor has to be proportional to the degree of confidences. Therefore, we use $w_{a_i} = 1 - D_i$. We fix $w_{a_i} = 0$ if the vertex p_i does not belong to the set of anchors. When the value of this weight is low, the ARAP deformation tends to maintain the differential geometry of the vertices, even if the final position of the anchors does not exactly match the target position. On the contrary, when this value is high, the deformation favors the position of the anchors given by the motion flow, despite the local rigidity prior. In our case, the weight of the anchors are kept to low values to deal with outliers and/or lack of precision from the motion flow. We consider the motion vectors like global informations which indicate the average direction and amplitude of displacements. However, we do not wish that the anchors vertices closely fit the indicated position but rather maintain the geometry of the mesh. As noticed in [27], the variation of the weight applied to the anchors leads to different deformation effects that can be expected. A low relative weight leads to a rough approximation of the target pose, preserving the surface details. Instead, a larger weight induces a strong position constraint which leads the surface to fit the target more accurately, despite the local rigidity prior.

4.4.3.2 Solving

The optimality condition for the minimum of our energy basically mirrors the result of [138], to which terms coming from the quadratic form (eq. 4.3) are added. That is, the optimal positions p' must satisfy:

$$\sum_{j \in N(i)} w_{ij}(p'_i - p'_j) + w_{a_i} p'_i = \sum_{j \in N(i)} \frac{w_{ij}}{2} (R_i - R_j)(p_i - p_j) + w_{a_i} p_i \quad (4.4)$$

where R_i is a local rotation best matching p_i and its one ring to p'_i . The global deformation is thus computed by iteratively solving a linear system and an optimal set of rotations matrices: we begin by computing the set of $\{p'_i\}_i$ which satisfy the optimal condition for a fixed set of initial rotations $\{R_i\}_i$ by solving a linear system of the form:

$$Lp' = b$$

where L corresponds to the Laplacian operator applied to the mesh M' in which we add the w_{a_i} weights related to each anchor point (eq. 4.3) on the diagonal, and b is a column matrix which contains the righthand side of eq. 4.3:

$$\begin{bmatrix} L + W \end{bmatrix} \begin{bmatrix} p'_{x_0} & p'_{y_0} & p'_{z_0} \\ p'_{x_1} & p'_{y_1} & p'_{z_1} \\ \vdots & \vdots & \vdots \\ p'_{x_n} & p'_{y_n} & p'_{z_n} \end{bmatrix} = \begin{bmatrix} b_{x_0} & b_{y_0} & b_{z_0} \\ b_{x_1} & b_{y_1} & b_{z_1} \\ \vdots & \vdots & \vdots \\ b_{x_n} & b_{y_n} & b_{z_n} \end{bmatrix}$$

with $b_i = \sum_{j \in N(i)} \frac{w_{ij}}{2} (R_i - R_j)(p_i - p_j) + w_{a_i} p_i$. Optimal rotations R_i are computed through singular value decomposition (SVD) from the positions of p_i and p'_i as derived in [138] (see appendix C.4). These two steps are repeated until convergence. Our convergence criterion is the difference between the energies $E_{ARAP}(M')_k$ and $E_{ARAP}(M')_{k-1}$ (see equation 4.2) computed at the current iteration k and the previous iteration $k - 1$, respectively. The iterative solving is stopped when the expression $E_{ARAP}(M')_k - E_{ARAP}(M')_{k-1}$ becomes greater than a predefined threshold. In our case, it is empirically set to $1 \cdot 10^{-4}$.

4.4.3.3 Implementation details

We used the *Eigen*¹⁸ library for matrix operations and system solving. The mesh processing is handled with *OpenMesh*¹⁹ library (which provides a *half-edge* mesh structure). A pseudo-code overview of energy minimization implementation is given in Algorithm 2. The list of anchor vertices and associated weights are computed as described in Section 4.4.2 (see Algorithm 1). The left part of the system is first build by computing the Laplacian matrix (see appendix C.5). The right part is then initialized with the positions of the vertices. For each iteration, we first compute the transformations matrix R (see appendix C.4) and update the X matrix afterwards. Functions are detailed in Algorithm 3.

```

Data: anchorList, anchorWeights, mesh
// build Laplacian 3D matrix from the mesh
 $\delta_{mesh} \leftarrow \text{computeLapMatrix}(\text{mesh});$ 
// compute a sparse Cholesky decomposition from Lap matrix ( $L \cdot L^T$ )
 $\text{LLT} \leftarrow \text{computeLLT}(\delta_{mesh});$ 
initB();
while  $\|D2 - D1\| > 1 * 10^{-4}$  do
    computeR();
    computeB();
     $D1 \leftarrow D2;$ 
     $D2 \leftarrow \text{computeD}();$ 
end
// copy final values of X in the mesh structure
for  $v_i \in \text{mesh.vertices}()$  do
     $v_i \leftarrow X.\text{row}(v_i.\text{index}());$ 
end
return ;

```

Algorithm 2: Iterative solver for energy minimization (see Algorithm 3 for methods' details.)

18. <http://eigen.tuxfamily.org>

19. <http://www.openmesh.org/>

```

initB()
    // init matrix B with mesh point values
    for  $v_i \in \text{mesh.vertices}()$  do
        for  $v_j$  in the neighborhood of  $v_i$  do
            B.row( $v_i.\text{index}()$ ) += ( $v_i - v_j$ );
        end
    end
    // add anchors
    for  $i \in \text{anchorList}$  do
        B.row( $v_i.\text{index}()$ ) += anchorWeights[i];
    end
    // init matrix X from B
     $X \leftarrow \text{LLT.solve}(B)$ ;
    return ;

computeR()
    // compute R
    for  $v_i \in \text{mesh.vertices}()$  do
         $R[v_i.\text{index}()] \leftarrow \text{computeR}(v_i)$ ;
    end
    return ;

computeB()
    // update matrix b
    for  $v_i \in \text{mesh.vertices}()$  do
         $r_i \leftarrow R[v_i.\text{index}()]$ ;
        for  $v_j$  in the neighborhood of  $v_i$  do
             $r_j \leftarrow R[v_j.\text{index}()]$ ;
            B.row( $v_i.\text{index}()$ ) +=  $(r_i + r_j)(v_i - v_j)$ ;
        end
    end
    // add anchors
    for  $i \in \text{anchorList}$  do
        B.row( $v_i.\text{index}()$ ) += anchorWeights[i];
    end
    // update matrix X
     $X \leftarrow \text{LLT.solve}(B)$ ;
    return ;

```

Algorithm 3: Iterative solver - methods' details

4.5 Local optimization

While the global deformation step of our approach provides a robust way to match the next pose, details of the pose due to non-rigid deformation (such as cloth folds or hair) are missed. A local optimization is thus required for the mesh to fit the new pose's silhouette. Moreover, mesh quality may also degrade over time as large deformation occurs, making mesh regularization desirable. We developed two different approaches to handle this local optimization.

4.5.1 First approach: numerical integration

The first solution we proposed uses a numerical integration where each vertex of the surface is considered as a solid object. Formerly, the global pose fitting may need to be adjusted as the local optimization cannot be efficient if some parts of the surface are too far from the target visual hull. The vertices are then regularized and closely fitted to the volume's surface with the numerical integration itself (see Section 4.5.1.2).

4.5.1.1 Adjustment

The selection of anchors' vertices (see Section 4.4.2) captures the largest motions between two frames. However, in several complex cases, the deformed template must be adjusted to match the pose of the next visual hull. Using the same anchors' selection described in section 4.4.2, we select a new subset of anchors corresponding to the most distant vertices according to the EDT values. These new anchors are pushed toward the visual hull, along the gradient vector computed on the EDT grid (see Chapter 3, Section 3.3). The anchors previously selected in the first ARAP deformation remain fixed. We then start a new ARAP deformation of the mesh, as described previously. This new deformation adjusts the global template position with the target pose defined by the visual hull. As the mesh deformation is more constrained and the displacements are noticeably shorter, this new deformation converges quickly. Depending on the character's motion complexity, several ARAP adjustments can be applied, until a global convergence between the target pose and the template is reached. Usually a maximum of two adjustment steps is enough.

4.5.1.2 Physics-based deformation

Several forces are applied on the vertices and the complete system is iteratively updated like in a mechanical system. We thus compute local vertex displacements based on both fitting accuracy and regularization as follows.

Regularization We regularize the mesh by applying spring-like forces to favor equal-length edges. We implement this term by considering a force per vertex of the type:

$$f_r(p_i) = \alpha \sum_{j \in N(i)} (\|p_i - p_j\| - \bar{r}_i) \frac{p_i - p_j}{\|p_i - p_j\|}$$

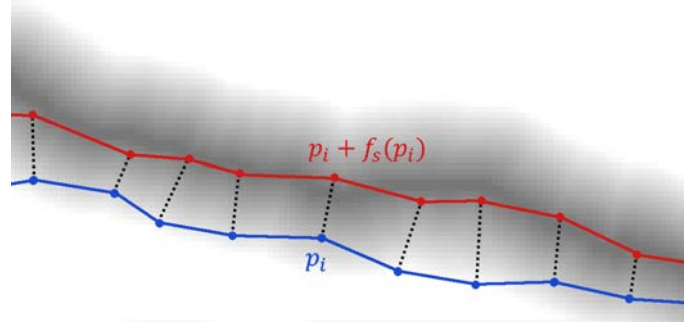


FIGURE 4.7 – **Silhouette fitting.** Vertices are moved along the local surface normal, according to the EDT values (grey levels).

where α is a fixed stiffness coefficient (we used $\alpha = 1$) and p_j is a vertex from the one-ring neighborhood of p_i , while the rest length \bar{r}_i is set to the current average length of the edges adjacent to p_i . We use only the tangential component of the resulting vector.

Silhouette fitting Using the EDT grid, we also push each vertex toward the visual hull surface by adding the following force:

$$f_s(p_i) = \sum_{j \in N(i)} (\mathbf{n}_{p_j} \cdot EDT(p_j))$$

with \mathbf{n}_{p_j} and $EDT(p_i)$ being the normal vector and the EDT value at p_i , respectively. As we only use the normal component of this vector (Figure 4.7), this expression can be seen as a simplified MLS projection described in [4].

Integration The resulting vectors f_r and f_s are added to obtain a displacement for each vertex. These two forces can be weighted, allowing the user to control regularization vs shape fitting depending on the noise present in the volume sequence. This displacement is integrated over 200 time steps between pose t_k and t_{k+1} by updating position and velocity of each vertex (assumed to be all of unit mass) using a simple Runge-Kutta (RK4) explicit integrator (see appendix C.6 for details).

4.5.2 Second approach: energy minimization

This second approach to the final optimization step is based on a global energy minimization. The process is similar to the first mesh deformation described in Section 4.3.3 except that the energy terms differ. The prior energy is still based on the ARAP equation. Indeed, the conservation of the differential coordinates is a better criterion than the regularization term in the previous approach. The data fitting is a quadratic energy minimizing the difference between the initial vertices' positions and target positions given by the data:

$$E_{DATA}(M') = \sum_{i=1}^n \|p'_i - p_i\|^2$$

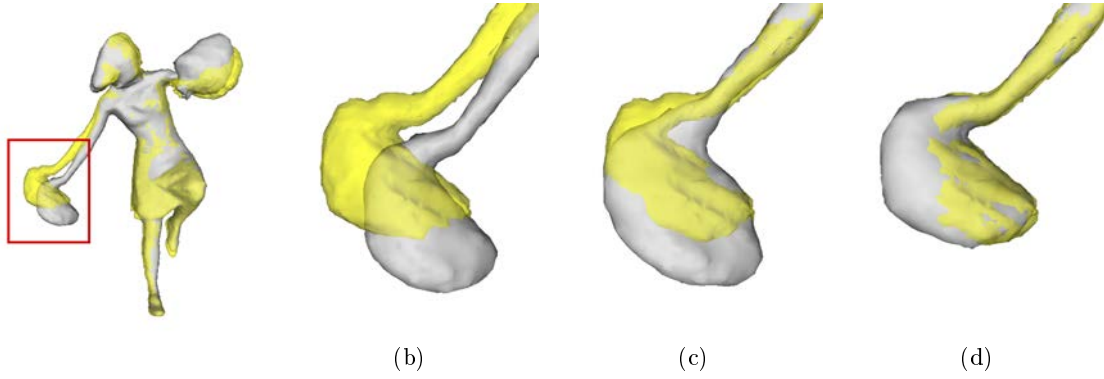


FIGURE 4.8 – **Local optimization.** Template mesh, figured in grey, at the initial pose (b) and target pose figured in yellow. Example of an inaccurate surface matching after the first mesh deformation following the motion vectors (c). A closer fitting is then obtained *via* the local optimization (d).

Note that this energy is similar to the anchor term described in equation 4.3 (see Section 4.4.3). However, it is this time applied to all vertices p_i in the mesh, instead of a reduced anchors' sample. The target position p'_i for each vertex is given by the EDT, as in the silhouette fitting term from the previous approach (see Section 4.5.1):

$$p'_i = \sum_{j \in N(i)} (\bar{\mathbf{n}}_{p_i} \cdot EDT(p_j))$$

where $\bar{\mathbf{n}}_{p_i}$ is the average normal vector of the neighborhood's vertices.

The final energy to minimize is the following:

$$E_{REG}(M') = \sum_{i=1}^n \sum_{j \in N(i)} w_{ij} \|(p'_i - p'_j) - R_i(p_i - p_j)\|^2 + w_{reg} \sum_{i=1}^n \|p'_i - p_i\|^2$$

This leads to the final system to solve:

$$\sum_{j \in N(i)} w_{ij}(p'_i - p'_j) + w_{reg}p'_i = \sum_{j \in N(i)} \frac{w_{ij}}{2}(R_i - R_j)(p_i - p_j) + w_{reg}p_i$$

where w_{reg} denotes a constant weighting term to determine the importance of silhouette matching in front of prior. A high value may induce important local deformations to fit the data, despite the differential coordinates' conservation. A low value leads to a rigid surface, with a high inertia. We used $w_{reg} = 1$ for a balanced influence of the two terms. This method has the advantage of maintaining the local rigidity prior and to avoid the ARAP adjustment required by the previous approach (see Section 4.5.1.1). The results are illustrated by Figure 4.8.

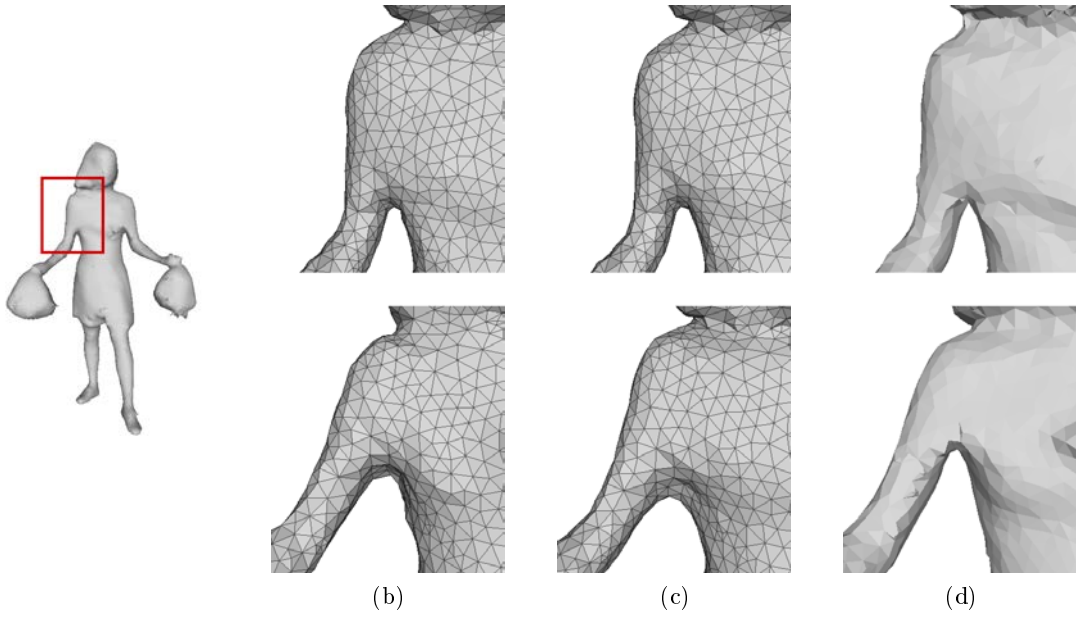


FIGURE 4.9 – Comparison between the two approaches of local optimization: numerical integration (b) and energy minimization (c) and target visual hull surface (d). Evolution of the mesh between the frames 56 (top row) and 61 (bottom).

4.6 Results

We tested our algorithms on the *short* and *long cheerleaders*, *astronaut*, *dancer*, and *capoeira* sequences (see Chapter 5, Section 5.2 for details). The as-rigid-as-possible global deformation (Section 4.4) usually needs between 300 and 1500 iterations to converge. It is performed in 430 iterations and 48s for the *short cheerleader*, 330 iterations and 28s for the *astronaut*, 360 iterations and 29s for the *dancer*, 600 iterations and 52s for the *capoeira*, and 1200 iterations and 63s for the *long cheerleader*. The number of iterations depends on the convergence criterion (see Section 4.4.3). It may also significantly vary from one pose to another and depends on the complexity and/or amplitude of the motions that occur in the scene. In our first tests with the numerical integration approach (see Section 4.5.1), we usually apply only two steps of ARAP adjustment which were performed in an average of 100 iterations. The local mesh optimization (Section 4.5.1.2) was applied with equal weights for the two forces f_r and f_s . An average of 200 iterations was necessary for the numerical integration. Due to the limitations that presented this approach, we finally used the second approach based on energy minimization (see Section 4.5.2) with equal weight applied to both terms of the energy (data fitting *vs* prior). The number of iterations for this optimization step also depend on the convergence criterion but is usually lower than 800 and are applied in an average of 100s.

Our mesh animation approach described in Section 4.4.3 leads to a locally rigid deformation, which preserves the mesh structure during the whole sequence. Figure 4.6 shows how the mesh is deformed, following the motion vectors, while maintaining local rigidity of the surface. This deformation prior brings the robustness compared to a simple advection of the vertices by the motion flow as described in Chapter 3. It should also be noticed that our use of weights based on the reliability of the anchors nicely extends the ARAP

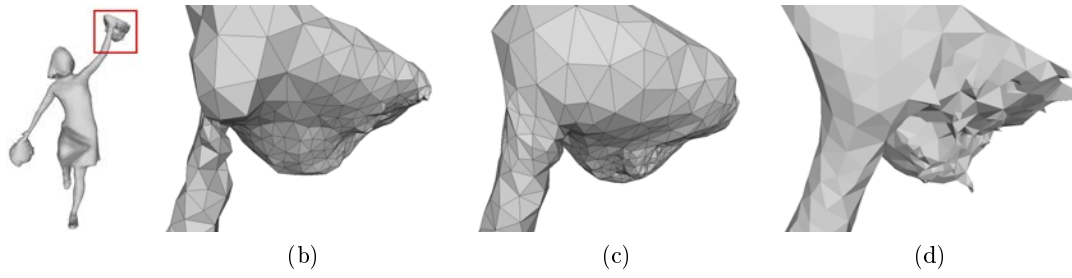


FIGURE 4.10 – Comparison between the two approaches of local optimization: numerical integration (b) and energy minimization (c) and target visual hull surface (d).

modeling technique, rendering it particularly robust to the inherent noise present in the motion flow. This improvement does not require higher computational costs since the added anchor energy (eq. 4.3) we proposed only adds diagonal elements in the Laplacian-like matrix involved in the original ARAP method.

Figure 4.9 shows one limitations of the energy minimization approach for the local optimization. On this example, the motion separates the arms from the torso (see the evolution of initial visual hull in Figure 4.9d). In this case, the optimization using energy minimization is limited in the adaptation of the surface to its new morphology (see Figure 4.9c). The numerical integration approach presents a better elasticity and allows the mesh to adapt (see Figure 4.9b). However it is not sufficient to ensure a consistent matching and often necessitates a preliminary ARAP adjustment (see Section 4.5.1.1). The numerical integration itself also need a timestep parameter which has to be fixed (see appendix C.6).

We also compare the two approaches proposed for local optimization (Section 4.5). The second method, based on energy minimization (Section 4.5.2) enables a deformation which preserves the local rigidity, such as the global pose estimation. It handles local displacements while avoiding the degradation of the local mesh's structure that happens with the first approach based on numerical integration (see Section 4.5.1). With this first method, the regularization applied on the edges helped cleaning the mesh's triangulation but could also lead to triangle stretching or unwanted vertex displacements (see Figure 4.10).

4.7 Conclusion

In this chapter, we have proposed a new approach for generating a time-evolving triangle mesh representation from a sequence of binary volumetric data representing an arbitrary, possibly complex and unstructured motion. Using the visual hull as a prior, we animate a template mesh, generated by a surface reconstruction of the first volume, *via* as-rigid-as-possible, detail-preserving transformations guided by the motion flow and based on a sparse set of weighted anchors. Several steps of this pseudo-rigid deformation can be applied to recover complex motions. A final local optimization adjusts the mesh to better match the mesh shape to the current visual hull, leading to a robust, temporally-consistent mesh reconstruction of the motion. Our approach assumes that the topology of the first frame of the input data is kept throughout the sequence. However, changes in the topology or morphology of the visual hull could occur in the captured sequence, possibly due to

occultations if not enough view angles are available. Currently, these changes are not supported by our approach. In the future, stereo-matching could be used to improve the accuracy and quality of the volume sequences. Alternatively, one could also handle topology changes through, for instance, the method proposed by Letouzey and Boyer [85]. Kravstov *et al.* [80] described a morphing of arbitrary meshes. However, this transformation is driven by an approximation of the shape described by an implicit function and based on a skeleton. In our case, as specified before, the shapes to be matched can hardly be represented this way, without a model prior. Bojsen-Hansen *et al.* [24] proposed a surface-tracking based on the non-rigid registration by Li *et al.* [90] and addresses the issue of topology changes by partially resampling the mesh, even if this operation, in our case, disrupts the continuity in the mesh's structure.

Chapter

5

Results

Contents

5.1	Introduction	99
5.2	Datasets	99
5.3	Results	99
5.3.1	Process flow and parameters	100
5.3.2	Computing times	101
5.3.3	Experiments and discussion	101
5.3.4	Mesh tracking measurement	104
5.4	Applications	107
5.5	Limitations	109
5.6	Conclusion	110

Résumé

Nous présentons dans ce chapitre les résultats obtenus suite à l'application de notre méthode. Nous détaillons dans un premier temps les jeux de données sur lesquels nous avons testé nos algorithmes (*cf.* Section 5.2) et les performances obtenues (*cf.* Section 5.3.2). L'ensemble du processus se déroule selon les étapes suivantes :

1. Initialisation du maillage *template* (*cf.* Chapitre 4, Section 4.4.1)
2. Estimation du champ de déplacements (*cf.* Chapitre 3, Sections 3.4 à 3.5)
3. Echantillonnage des points d'ancrage (*cf.* Chapitre 4, Section 4.4.2)
4. Déformation globale du maillage de référence vers la pose suivante (*cf.* Chapitre 4, Section 4.4.3)
5. Optimisation locale de la surface (*cf.* Chapitre 4, Section 4.5.2)
6. Répétition des étapes 2 à 6 jusqu'à la fin de la séquence initiale

Le résultat final est une nouvelle succession de poses, cette fois-ci définies par un maillage triangulé dont le nombre de sommets et la connectivité restent constants (maillage dynamique). Afin de mesurer la qualité du suivi des poses reconstruites par notre surface dynamique, nous mesurons à chaque *frame* la distance entre l'enveloppe visuelle et le maillage *template* à la pose correspondante (*cf.* Section 5.3.4). Les résultats montrent que le maillage animé que nous obtenons suit correctement la succession de poses décrites par les séquences d'enveloppes visuelles.

Nous détaillons ensuite la manière dont ces résultats peuvent être exploités dans le cadre d'applications de post-productions (*cf.* Section 5.4). Nous utilisons le format Alembic pour exporter le maillage animé vers des logiciels d'animation 3D. Le personnage reconstruit peut ainsi être placé dans une scène virtuelle. La cohérence temporelle qui caractérise les maillages permet de suivre la position des sommets tout au long de la séquence. Ainsi, les contacts entre la surface animée et des objets 3D de la scène peuvent être détectés à tout moment lors de l'animation. Le personnage reconstruit peut de cette manière interagir avec l'environnement virtuel (collision avec des objets et/ou des particules, accessoire virtuel ... *etc.*). La triangulation constante du maillage facilite également l'application de textures à partir des images caméras car elle permet de calculer une *UV-map* unique pour l'ensemble de l'animation. La qualité visuelle de la reconstruction est également améliorée en faisant disparaître des effets de *flickering* dus aux artefacts de la reconstruction multi-vues initiale.

Enfin, nous discutons des limites de notre méthode. Le critère de cohérence temporelle que nous nous imposons rend difficile la gestion des séquences qui contiennent des changements de morphologies entre deux poses consécutives des séquences reconstruites. De tels événements pourraient être traités en ayant recours à un ré-échantillonnage partiel du maillage, limité aux zones de contact, ce qui peut également entraîner des changements topologiques de la surface.

5.1 Introduction

In this chapter, we present extensive, comparative results, of our mesh animation method. After the computation of the motion flow in Chapter 3, a template mesh is deformed following the displacement vectors to match the successive poses from the input reconstructed sequences. This mesh deformation, described in Chapter 4 uses, for each frame of the animation, a selection of anchor vertices whose motion is driven by the motion flow. These anchors then act as constraints to lead the deformation of the complete template surface. This inter-frame displacement leads to the next pose of the input time-series. This process is repeated throughout the sequence until the template mesh has fitted all the successive poses (see Chapter 3, Figure 3.1).

After a detailed description of the datasets we used to test our algorithms (see Section 5.2), we present the final animation we obtained (see Section 5.3). We also discuss the properties and noticeable components of these results. The performance and behavior of the algorithms described in this manuscript are also discussed, as well as the impact of parameters and comparison with previous work. In Section 5.4, we propose several concrete applications for our results and show how our time-consistent meshes handle several drawbacks in the exploitation of multi-view reconstruction technologies for post-production processing. Finally, we discuss limitations of our approach and propose several improvements.

5.2 Datasets

We evaluate our method on several datasets obtained through volumetric visual hull reconstruction (see Table 5.1). The *short cheerleader*, *long cheerleader*, and *astronaut* sequences come from an indoor studio shoot using the 24-camera rig of the RECOVER 3D studio (see Chapter 1, Section 1.3.2). The *Simon* sequence was simulated instead of captured (*i.e.* reconstruction of a digital character with a virtual cyber-dome software). The *dancer* sequence was generated using the multi-viewpoint images provided by the GrImage platform²⁰. Due to the low number of cameras (8 viewpoints) and the low resolution of the pictures, this dataset produces coarse visual hulls. We also tested our method on the *capoeira* sequence, using the multi-view videos described in [45]²¹. This reconstruction also suffers from the reduced number of viewpoints (8 cameras too). Due to several inconsistencies in the silhouettes' segmentation, the visual hulls contain many artifacts (holes and phantom volumes, see Figure 5.2) All timings were measured on a 64 bit Intel Core i7 CPU 2.20 GHz. Results from these sequences are presented in Figures 5.1 and 5.2, demonstrating the robustness of our approach despite the coarseness of the input volumes.

5.3 Results

We now discuss the results we obtain with the datasets previously described. We first summarize the complete process flow of our method (see Section 5.3.1) and the perfor-

20. <http://4drepository.inrialpes.fr/>

21. <http://resources.mpi-inf.mpg.de/siggraph08/perfcap/>

Name	Number of frames	Average resolution	Number of vertices (template)	Source images
<i>Short cheerleader</i>	25	$180 \times 270 \times 170$	5158	RECOVER 3D studio capture
<i>Long cheerleader</i>	220	$180 \times 295 \times 160$	9619	RECOVER 3D studio capture
<i>Astronaut</i>	25	$150 \times 120 \times 330$	8048	RECOVER 3D studio capture
<i>Simon</i>	50	$114 \times 259 \times 95$	5158	Synthetic
<i>Dancer</i>	30	$150 \times 100 \times 300$	7843	GrImage studio capture
<i>Capoeira</i>	40	$150 \times 100 \times 300$	7774	Perfcap studio capture

TABLE 5.1 – Datasets.

mances of our algorithms (see Section 5.3.2). We then discuss the visual quality of the results (see Section 5.3.3) and complete these observations through a distance measurement between the output animated surface and the input reconstructions (see Section 5.3.4).

5.3.1 Process flow and parameters

The complete process used to generate a time-consistent mesh animation comprises the following steps:

1. Extracting the template mesh from the first pose ($t = 1$) of the input sequence (see Section 4.4.1). A manual cleaning of the vertices may be necessary when the model-free reconstruction (visual hull) produces inconsistent artifacts (holes or phantom volumes).
2. Compute the Poisson disk sampling on the template mesh (see Section 4.4.2).
3. Compute EDT grids for volumes t and $t + 1$ (see Section 3.3).
4. Compute the motion flow between poses t and $t + 1$ (see Sections 3.4 and 3.5). If $t > 1$, this computation is speed-up using the prediction described in Section 3.6.
5. Select anchor vertices (see Section 4.4.2).
6. Apply global pose deformation of the mesh (see Section 4.4.3).
7. Apply local optimization (see Section 4.5.2). The template mesh then corresponds to pose $t + 1$.
8. $t = t + 1$, return to step 3 until the end of the sequence.

Several parameters must be user-specified. Most of them vary with the input data (especially the resolution of the voxel grid and the amplitude of the motions). We describe here the values we empirically fixed for our datasets. The Poisson sampling is processed with a distance criterion (disk radius) equals to $5 \times V$, with V being the size of a voxel.

Dataset	Total inter-frame processing	Motion flow computation step	Mesh deformation step
<i>Short cheerleader</i>	198	37	161
<i>Long cheerleader</i>	203	75	128
<i>Astronaut</i>	182	59	123
<i>Dancer</i>	171	49	122
<i>Capoeira</i>	395	248	147

TABLE 5.2 – Average inter-frame computing time in seconds.

We also set a maximum threshold to the number of sampled vertices: 20% of the total number of vertices. The voxel matching (see equation 3.3) is computed with the following weights: $\omega_p = 0.3$, $\omega_n = 0.45$ and $\omega_c = 0.25$. The motion flow regularization (see Section 3.5) is applied with 7^3 voxels windows centered of the voxel to process. During the anchors' selection step, 5% of the mesh vertices are segmented (3% associated with minimal EDT values and 2% for maximal values, as described in Chapter 4, Algorithm 1). The number of iterations for mesh processing are not fixed, the iterative solver automatically stops when the convergence criterion is reached (see Chapter 4, Algorithm 2). The energy of the local optimization step (see Section 4.5.2) is computed with an equal weight for E_{DATA} and E_{REG} .

5.3.2 Computing times

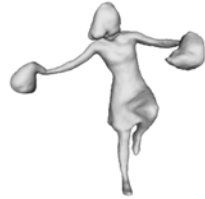
The performance of the algorithms we used for the separate steps of our method are detailed in preliminary results of Chapter 3 (see Section 3.7) and Chapter 4 (see Section 4.6) for motion flow computation and mesh deformation, respectively. Table 5.2 lists the average computation time required to apply the complete process flow, summarized in Section 5.3.1, applied between two consecutive frames of the sequences. Note that we did not use parallelized and/or GPU implementations. All timings were done on a 64-bit Intel Core i7 CPU 2.20 GHz. Note that times are average values over the complete sequences and are specific to our datasets. The computing time required for motion computation and the number of iterations for mesh animation may vary a lot, depending on the volume resolution, the displacements' amplitude and speed, and the complexity of motions and surfaces.

5.3.3 Experiments and discussion

The *cheerleader* dataset shows that the shape of the pom-poms is correctly adjusted after the template deformation (see Figure 5.1). Our mesh animation method leads to an adaptation of the template during the sequence, avoiding some of the model-based inconveniences, as in [45], where the tracked model retains some of the surface details (clothing folds) from the initial pose during the whole sequence. With the *dancer* sequence, we show that the mesh correctly tracks the shape of the moving dress (Figure 5.3). This type of animation would be hardly recovered with an articulated model tracking approach.



Cheerleader (short) visual hull sequence



Cheerleader (short) mesh animation



Astronaut visual hull sequence



Astronaut mesh animation

FIGURE 5.1 – Results of the mesh animation for the *cheerleader (short)* and *astronaut* sequences. Comparison between the original visual hull reconstruction (top) and the temporally consistent mesh (bottom).

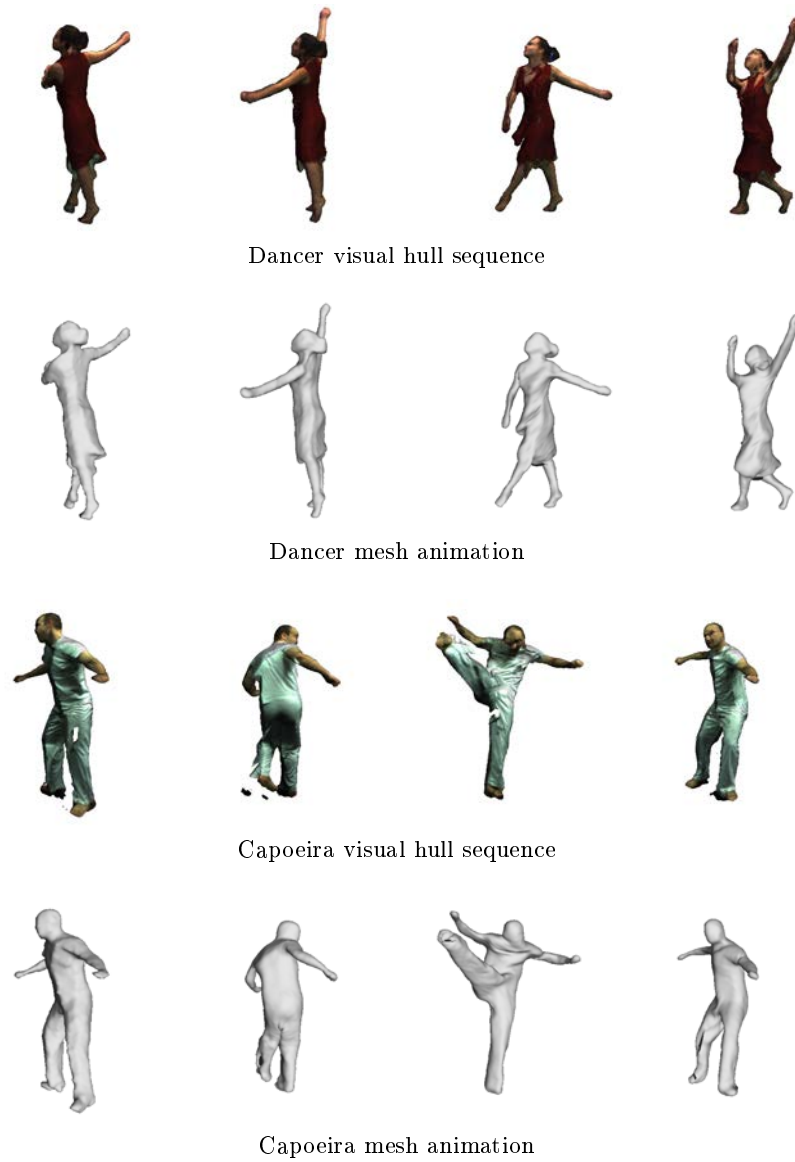


FIGURE 5.2 – Results of the mesh animation for the *dancer* and *capoeira* sequences. Comparison between the original visual hull reconstruction (top) and the temporally consistent mesh (bottom).

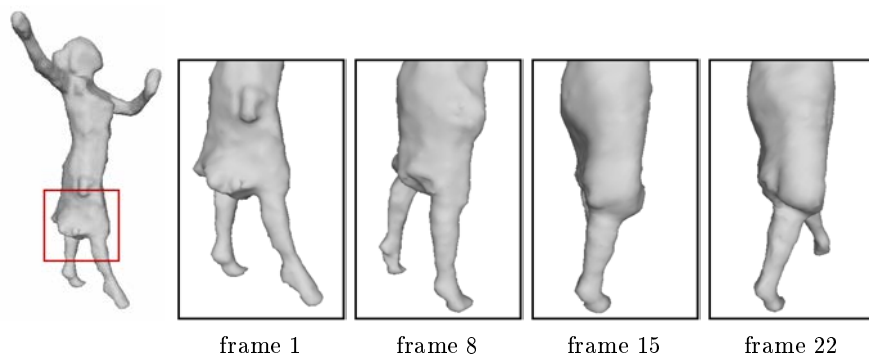


FIGURE 5.3 – Example of free-moving clothes' tracking in the *dancer* sequence.

In the *Simon* sequence, we encountered problems to handle thin details, as described in Chapter 3 (see Section 3.7.4). Moreover, collisions between the legs (causing large alterations in the morphology of the shape and modifying the topology) and lack of texture lead to inaccurate results. Other sequences provide consistent mesh animations which quality is evaluated in the next section (see Section 5.3.4).

5.3.4 Mesh tracking measurement

The quality of the template fitting with each pose of the input sequence is measured by computing the distance between the visual hull surface and the deformed template. We illustrate this measurement in Figure 5.4 with several poses of the *astronaut*, *dancer*, and *capoeira* sequences and show that the matching quality stays almost uniform on the complete surface during the animation. The quality of the matching strongly depends on the quality of the visual hull (*e.g.*, the extrema on the *astronaut's* right leg at frame 24 corresponds to a hole in the input visual hull). The *capoeira* sequences is composed by very noisy visual hulls which contains many holes, artifacts and outlying voxels (due to the low resolution on the images and the limited number of viewpoints), especially on the legs (see Figure 5.2). This explains why the shape of the feet quickly disappears. This sequence is also characterized by fast motions which induce large inter-frame displacements. However, our method maintains a consistent shape tracking during the 25 poses.

The Figure 5.5 shows the same result for the *long cheerleader* sequence. The colors represent a signed distance (negative inside the object, positive in the outside – we used the CloudCompare²² implementation), for each vertex of the deformed template, to the visual hull surface. The values are expressed in the same unit as the vertices coordinates and are included between a minimum of -94.43 and a maximum of 125.20. Extremum values are figured in blue (negative) and red (positive), whereas zero (minimal distance) is between green and yellow. It can be noticed that these extremum values are mostly located on the pom-pom which are animated by free motions of high amplitudes. Note also that, even if our mesh structure stays consistent during the complete animation, the wrists' and hands' surfaces slowly degrade over the animation. These parts of the surface undergo important and complex motions during the animation and also correspond to thin sections of the visual hull regarding to the voxels' size. These properties make the surface tracking particularly difficult for this part of the mesh. A better voxel resolution could handled these limitations.

With the *capoeira* sequence, the lower quality of the multi-view images and silhouettes produces a noisy and damaged reconstruction. The visual hulls of this sequence contain many inconsistencies such as occlusion artifacts and holes in the character's shape. Our tracking method, based on visual hull matching, is thus hardly relevant for long sequences from this dataset. As the video capture technologies quickly evolves toward high quality pictures and stereo carving, producing accurate model-free reconstruction, we should reduce this type of artifacts. Note that the RECOVER 3D project also aims at reconstructing high quality volumes, computed by high resolution visual hulls enhanced with stereo-matching. Nevertheless, we note that our template animation ensures that our mesh matches the silhouettes better (Figure 5.6), whereas the original method described by de Aguiar *et al.* [45] is closer to a motion capture approach, with a high quality template

22. <http://www.danielgm.net/cc/>

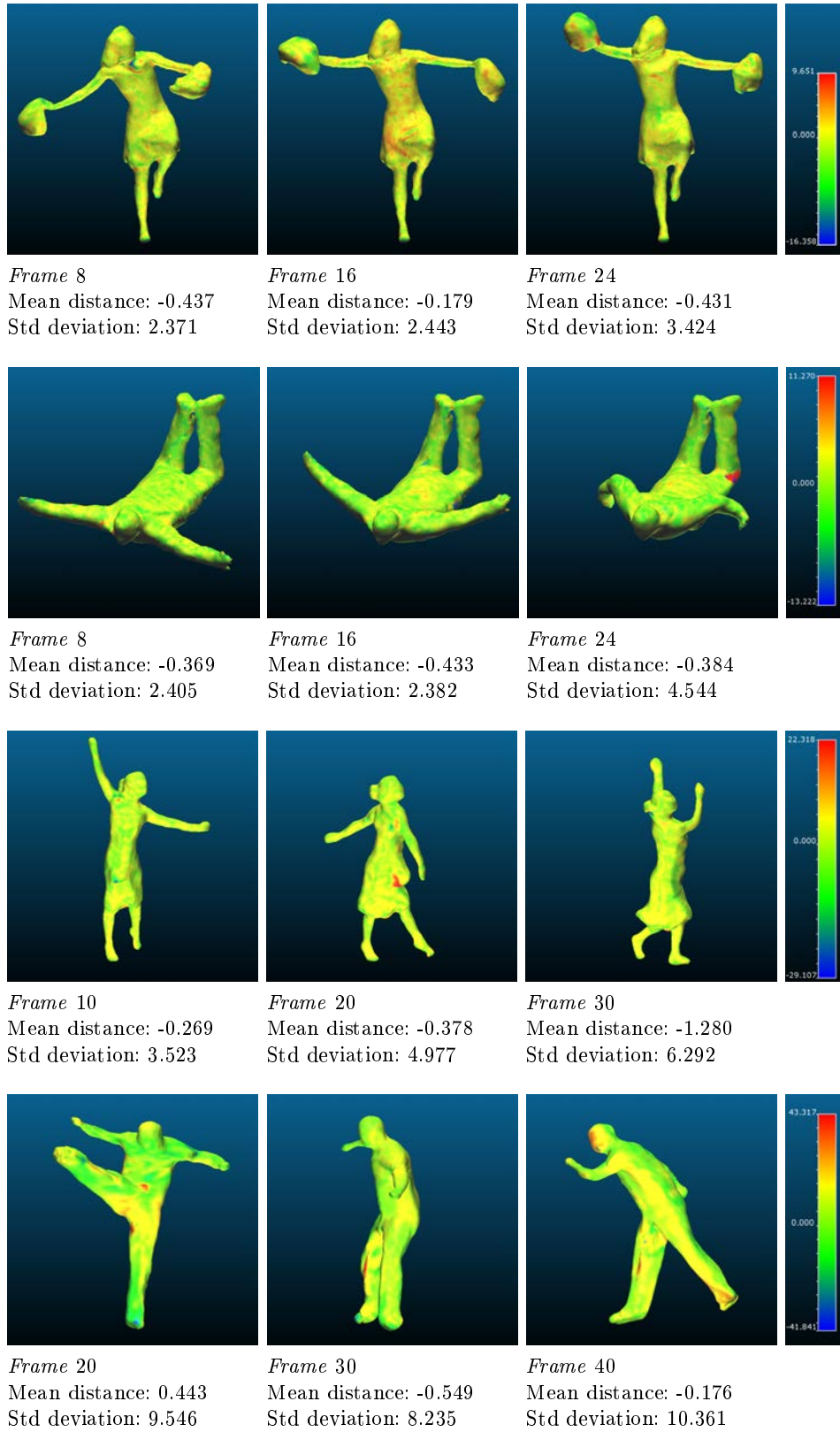


FIGURE 5.4 – Surface matching measurement for the *short cheerleader*, *astronaut*, *dancer*, and *capoeira* sequences.

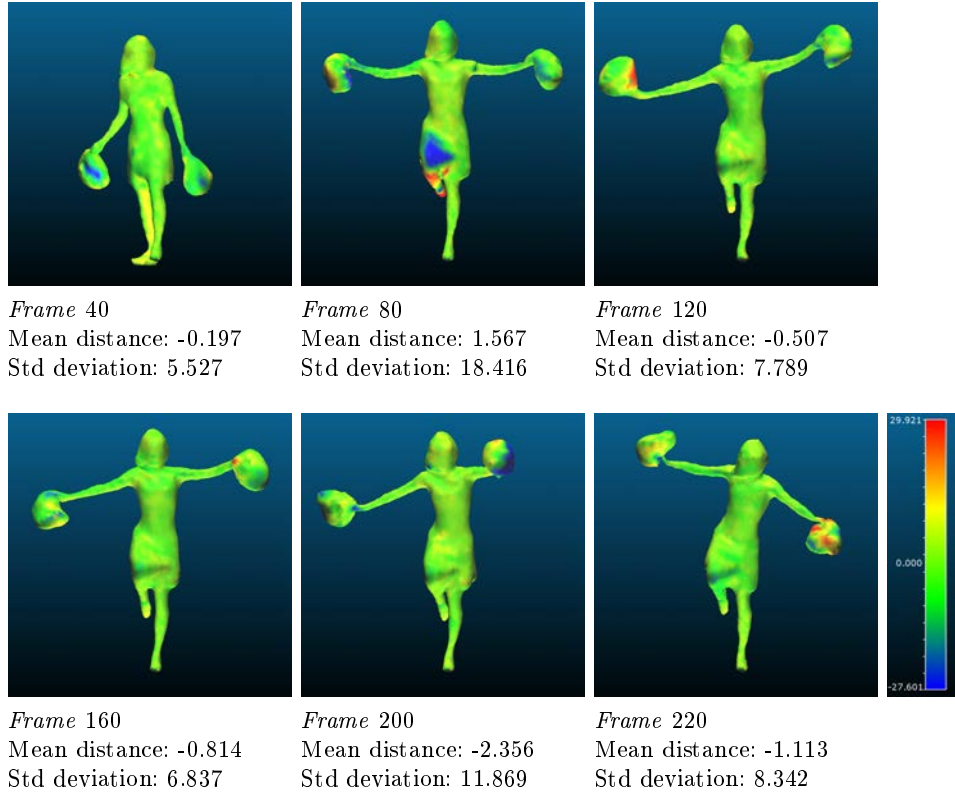


FIGURE 5.5 – Surface matching measurement for the *long cheerleader* sequence.

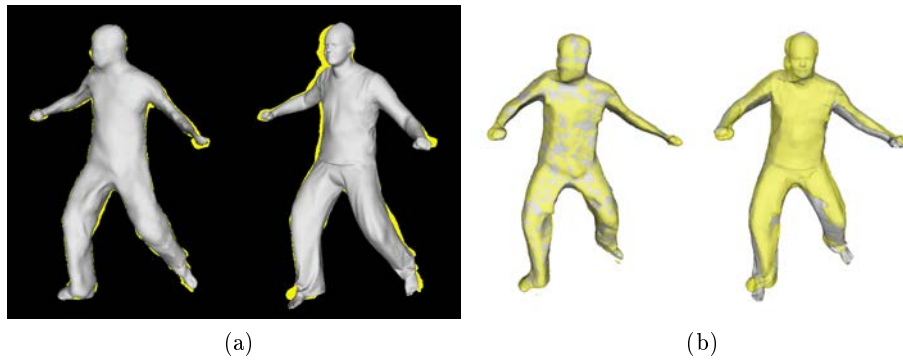


FIGURE 5.6 – **Capoeira sequence, comparison with [45].** (a) silhouette overlap. (b) Comparison between visual hull (yellow) and 3D mesh (grey). Our method (left), despite the low quality of the visual hull, matches the silhouette better than the model-based method proposed in [45] (right).

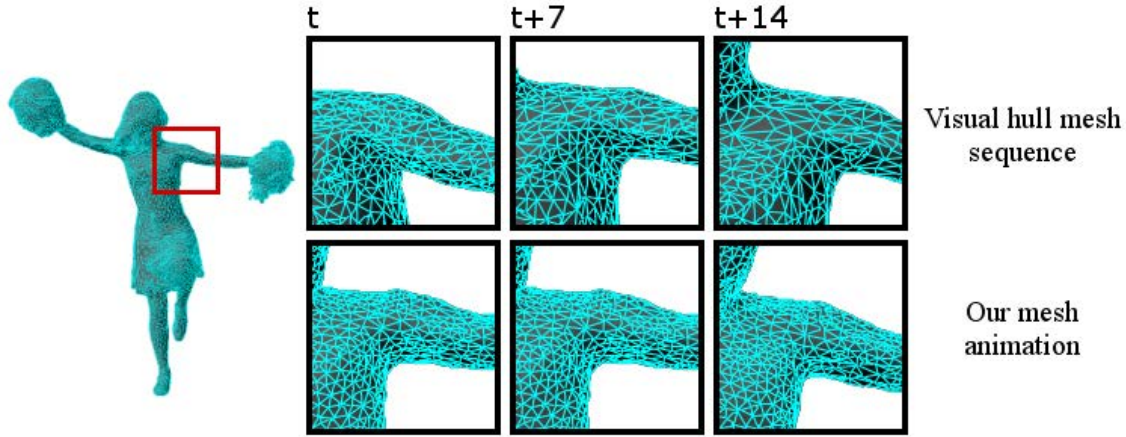


FIGURE 5.7 – Comparison between the temporally inconsistent mesh sequence from a model-free reconstruction (top) and our animated mesh (bottom).

animated by picture-based constraints and roughly matching the actor’s pose but without adaptation to the tracked surface, such as the clothes’ deformations. The average Hausdorff distance computed (with respect to the visual hull’s bounding box) between the visual hull of the target pose and the animated template resulting of the model-based method [45] is 0.011933 whereas we obtain an average of 0.003291 with our approach.

5.4 Applications

Our temporally coherent character can easily be placed in a virtual environment as a simple mesh animation. Such virtual cloning system allows us to use the reconstructed actor like a 3D animated character. A virtual camera can then be used with free viewpoints and trajectories, without being limited by the physical properties of the shooting studio. The rendering of the virtual scene is noticeably easier with this animated object than with mesh sequences where each pose of the sequence must be loaded before the rendering of the corresponding frame. We directly export our mesh animations to the Alembic file format²³, accepted by widely used software such as Autodesk Maya or RealFlow. The temporal consistency of our output dynamic mesh, meaning that the vertices can be followed during the entire animation (Figure 5.7), allows to compute dynamic interactions with the virtual environment, like collisions with objects or particles, as shown in Figure 5.9. Generating an output mesh for which only mesh vertices evolve in time (Figure 5.7) has multiple advantages for subsequent media production. First, it noticeably reduces the flickering effect of visual hull reconstruction, leading to a better visual quality. Second, vertices can be used to anchor virtual accessories (virtual makeup for instance). Third, collision with virtual objects (clothes or other) and environment is easy to detect and handle as one can rely on temporal coherence of the vertices of the output mesh, as shown in Figure 5.9. Finally, the animated mesh can keep the same texture during the animation, instead of computing a new texture for each frame. For long sequences, it is better to conserve a single UV map for the whole sequence, associated with an animated texture (in our case, we use a LSCM [87] texture atlas, see Figure 5.8) to prevent a visual texture *sliding* effect.

23. <http://www.alembic.io>

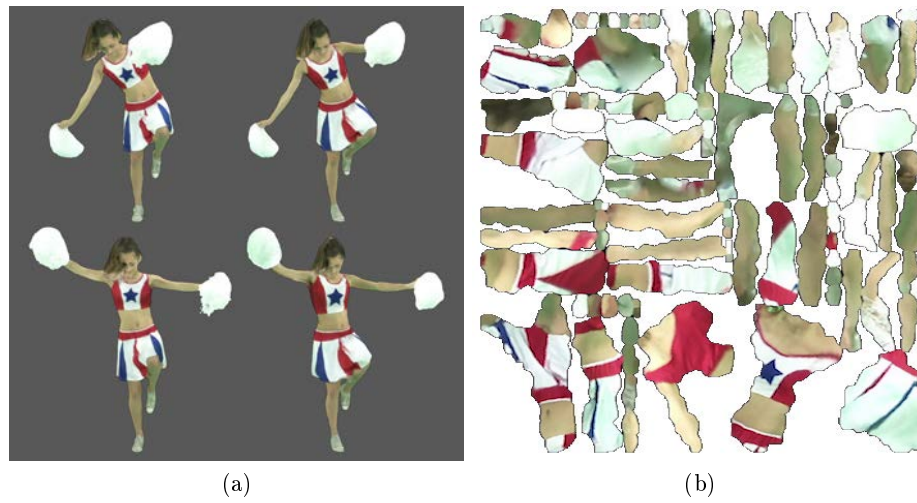


FIGURE 5.8 – **Applications.** (a) Comparison between the visual hull textured mesh sequence (left column) and our animated template with a single texture (right column). (b) Time-consistant mesh unwarping for UV mapping.

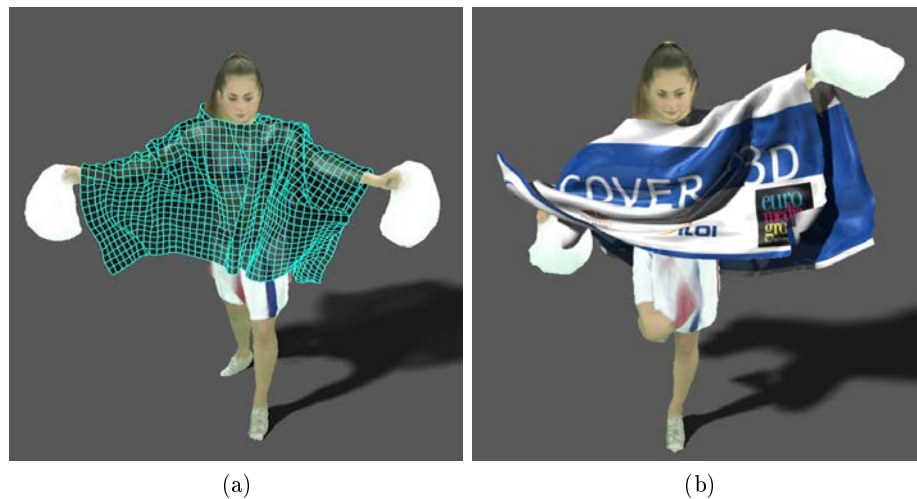


FIGURE 5.9 – **Applications.** Example of interaction (collisions) with a virtual coat. Wireframe (a) and textured (b) rendering of the virtual object, added to the reconstructed character.

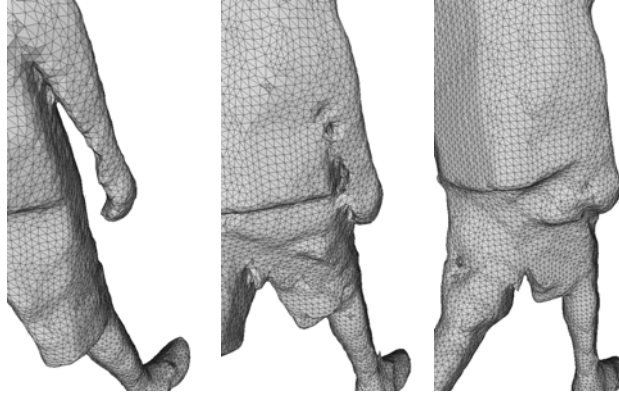


FIGURE 5.10 – **Limitation.** Example of morphology modification between several consecutive poses of a visual hull sequence.

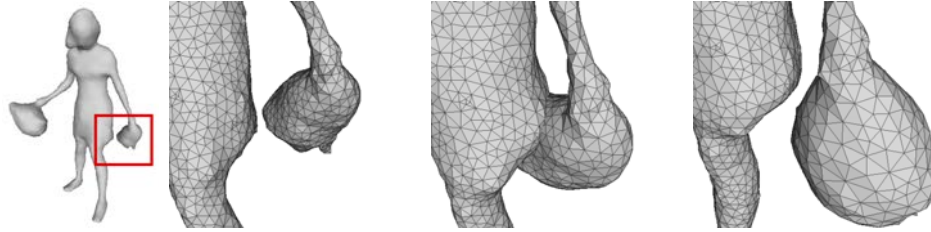


FIGURE 5.11 – Example of correctly handled surface collapsing during the mesh animation.

5.5 Limitations

The main limitation of our method is that the output animated model cannot handle any change in the topology of the surface. Some important modifications in the morphology of the visual hull may also not be supported. The Figure 5.10 shows an example of such case where the arm of the actor collapses with the body. The improvement of the initial reconstruction of the sequences could limit the occurrence of these problems, when it is only caused by occultation, if the visual hull is accurately carved to recover concavities. It should also be noticed that our system can still support the collapsing of two sections of the mesh if it does not imply an important modification of the morphology and if it is limited in time. Figure 5.11 shows an example during the *long cheerleader* sequence which is correctly handled by our system. These limitations could be handled through, for instance, the method proposed by Letouzey and Boyer [85]. Kravstov *et al.* [80] described a morphing of arbitrary meshes. However, this transformation is driven by an approximation of the shape described by an implicit function and based on a skeleton. In our case, the shape to be matched could be hardly represented in this way, without a model prior. Bojsen-Hansen *et al.* [24] proposed a surface-tracking based on the non-rigid registration by Li *et al.* [90] and addresses the issue of topology changes by partially resampling the mesh, even if this operation, in our case, disrupts the continuity in the mesh's structure.

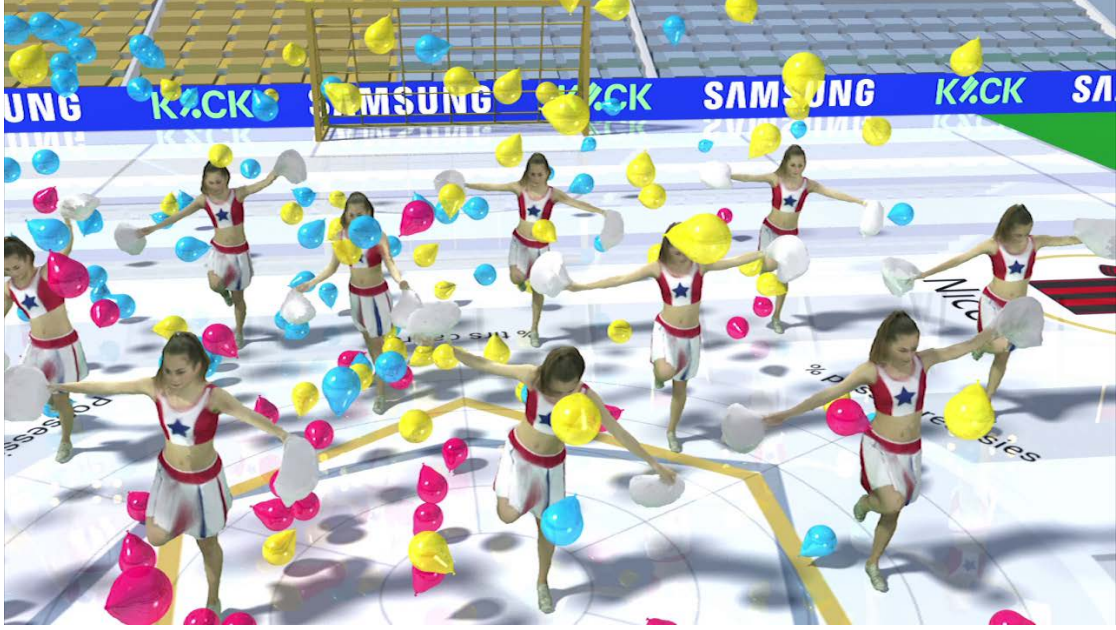


FIGURE 5.12 – Virtual crowd composed by several cloned (and temporally shifted) *cheerleader* animations in a virtual scene, interacting with a computer generated physical simulation of baloon particles.

5.6 Conclusion

This chapter described the final results we obtained with our complete process flow, that turns input visual hull sequences into time-consistent mesh animations. Our method is effective on various datasets. We also discussed the advantages our approach brings for post-production applications of multi-view reconstruction. The main advantage is the possibility to track the vertices of the mesh over the animation, which enable to compute interactions between the reconstructed character and the virtual environment. This process flow is adapted to the offline mode of the RECOVER 3D framework (see Section 1.3). Our method still suffers from limitations. However, future improvements of the initial model-free reconstruction should provide more accurate input data and thus lead our approach to a more precise pose-tracking.

Chapter 6

Conclusion

Contents

6.1	Summary	115
6.2	Future work	116
6.2.1	Short-term	116
6.2.2	Long-term	117

Résumé

Ce chapitre de conclusion dresse le bilan des travaux de recherche menés dans le cadre de cette thèse. L'approche que nous avons proposée apporte des contributions sur deux étapes principales: le calcul d'un champ de mouvements à partir d'une succession de poses statiques reconstruites et l'animation d'une surface de référence guidée par les vecteurs de ce champ de mouvements. Notre approche est entièrement automatique et présente l'avantage de ne pas être limitée à une morphologie prédéfinie. De plus, notre modèle de surface dynamique peut être animé de mouvements libres et n'utilise pas de structure articulée pour guider le déplacement des sommets. Par ailleurs, nos résultats respectent la contrainte de continuité temporelle fixée à l'origine de nos recherches. Les personnages virtuels que nous produisons sont représentés sous la forme d'un maillage unique dont la connectivité reste constante tout au long de l'animation. De cette manière, seules les positions des sommets évoluent au cours du temps. Cette caractéristique permet de calculer des interactions entre le clone numérique de l'acteur et son environnement virtuel, comme expliqué dans le Chapitre 5 (*cf.* Section 5.4).

Nous détaillons également dans ce chapitre les travaux futurs qui pourront être conduits à la suite du projet RECOVER 3D (*cf.* Section 6.2). Les améliorations à court terme concernent essentiellement le traitement de volumes de haute résolution. En effet, le système de reconstruction actuel est amené dans un avenir proche à effectuer des reconstructions de plus grande précision, enrichies notamment grâce à la fusion des approches silhouettes et stéréo. De plus, les caméras actuelles du *cyber-dôme* pourraient être remplacées par les caméras 4K (*UHD* 3840×2160) afin de capturer des images de meilleure qualité. Ces améliorations nécessiteront de travailler sur des volumes de résolution équivalente ou supérieure à 1024^3 voxels et nous mèneront à développer des structures de données optimisées. D'autres pistes sont également envisageables quant à la poursuite de ces travaux à long terme. L'évolution actuelle des technologies de capture et de reconstruction multi-vues mène à la utilisation de studios de plus en plus vastes. Dans ce cadre, un volume de capture de plus grandes dimensions permettrait de reconstruire plusieurs acteurs simultanément, ainsi que des éléments de décors. Il est donc envisageable d'améliorer notre approche actuelle de manière à effectuer le suivi de plusieurs acteurs dans une scène ainsi que leurs interactions avec des objets réels. Enfin, notre approche reste sensible aux changements qui apparaissent dans la topologie et/ou la morphologie de l'acteur entre deux poses consécutives. Actuellement, notre critère de cohérence temporelle nous empêche de modifier la connectivité du maillage lors de l'animation. De telles modifications sont cependant nécessaires si nous souhaitons traiter ce type de scènes complexes.

6.1 Summary

In this thesis, we described a new method to generate an animated 3D character from time-series of volumetric poses of an actor. We addressed several steps to obtain a time-evolving mesh, dealing with the constraints of an industrial multi-view reconstruction framework. Our method is inspired by both model-based and model-free approaches but is not limited by any restriction about the type of character and motion to reconstruct. The complete process is designed to handle any type of shape, including actors wearing costumes. This constraint prevented us to use any type of articulated model commonly used in various kind of 3D virtual cloning of actors. These specificities allow to keep the efficiency of multi-view reconstruction where actors are directly modeled in digital 3D models from a non-invasive indoor video capture. After a review of state-of-the-art techniques to produce time-consistent models from multi-viewpoint video captures, we detailed our approach.

First, we developed a motion flow computation that gives an estimation of the displacements of the captured actor between two adjacent frames of the input volume sequence. This method is adapted to our volumetric input which are produced by a visual hull reconstruction using a voxel-carving method. We compute an initial set of correspondences between the two shapes with a voxel matching algorithm. Next, the resulting 3D vectors field is filtered to obtain a regular motion flow. This process is repeated throughout the successive poses of the sequence to recover the complete motion of the actor. Even if this approach is adapted to our specific input, it is not limited by any assumption on the content of the reconstructed scene and allows to work on various types of characters.

Secondly, we described a mesh deformation process to match a template surface, with the successive poses of the sequence. This deformation is driven by the motion flow extracted in the previous step. Our 3D mesh registration follows a local rigidity prior and is performed in several steps. A set of anchor vertices are first sampled on the template mesh. These anchors are then displaced according to the vectors of the motion flow and thus drive the deformation of the complete surface through a global energy minimization. Once the new shape of the template surface roughly matches the target pose, a local optimization is performed. This last step handles surface details evolution and adjusts the mesh to closely fit the volumetric visual hull. The new pose of the initial mesh is then used as template to perform the deformation toward the next pose, until the end of the sequence. This type of as-rigid-as-possible mesh processing is less constrained than usual articulated priors and allows to deal with free-form animations while maintaining the mesh consistency thanks to a local rigidity criterion.

In the last part of this manuscript, we presented our results and the way they are used in post-production applications. Our method has been proven efficient on various datasets, generated from images captured with the RECOVER 3D studio captures as well as other multi-view reconstruction infrastructures. We show that our time-consistent models can be exported to 3D animation software so they can be directly included in a virtual environment like any hand-made animated character. Our model has several advantages in front of the mesh sequences used until now. It first improves the quality of the surface and significantly reduces the flickering effects of mesh sequences rendering. The main advantage we provide is the constant connectivity of the mesh, which means that the vertices are tracked over the complete animation, allowing to compute collision and/or follow their trajectories. This way, the reconstructed character can interact with virtual objects. It also simplifies

the texture mapping of the model.

6.2 Future work

6.2.1 Short-term

The research work lead in the RECOVER 3D projects is focused on two main innovations. The first one is the construction of temporally consistent models described in this thesis. The second is the amelioration of the model-free reconstruction processed, which should be both used for live onstage reconstruction of actors (online mode, see Section 1.3) and for input data to our method (offline mode).

6.2.1.1 High resolution volumes

At first, the volumetric visual hulls should be constructed with a higher voxel resolution (512^3 to 1024^3 3D grids). These grids could not be processed without an adapted data structure. Optimized grid models, such as *octree* structures, could be used to handle these bigger volumes. It should also be noticed that most of the data we use in our approach are sparse: the surface voxels in the original volumes and the 3D vectors in the motion flow are only defined in a reduced amount of voxels. Therefore, a grid structure that does not allocate memory for these void squares should be appropriate. In the same way, we can also notice that the EDT values are mainly used for the voxels which are close to the surface of the visual hull. Indeed, the access the EDT voxels to compute the surface voxels' normal vectors (see Section 3.4.2), select the anchor vertices (see Section 4.4.2) and to guide the local optimization (see Section 4.5.2). In most of these cases, the voxels we access are located in a small neighborhood of a surface voxel, except in the case of large motions where the anchors sampling needs to access to a EDT value far from the surface. Therefore, the EDT could also be optimized with a data structure of non regular voxels' size where the precision decreases proportionally with the distance to the surface. However, if these type of volumes' representation could be handled using data structures such as *hash maps*, it may also imply a significantly higher computation time due to the more complex access to the grid's elements.

6.2.1.2 Multi-view reconstruction improvements

The RECOVER 3D infrastructure also expects to use the multiscopic camera units to perform stereo-based reconstruction (see Section 1.3). Several depth-maps will be computed from several viewpoint surrounding the actor. These depth map could be used to carve the voxels of the visual hull. This approach also implies a high voxel resolution to produce a fine reconstruction which includes thin details of the surface. Other approaches for mixing several depth-maps could produce 3D meshes of point-clouds. In any case, our method will have to be adapted to these input. Currently, the RECOVER 3D studios uses HD cameras which produce images of 1920×1080 pixels. However, these cameras could be replaced, in a near future, by 4K cameras (UHD 3840×2160 or more). This improvement could allow to compute more detailed reconstructions but may require to increase the voxel

grid's resolution. These future improvements (stereo-matching and high resolution images) combined will offer the possibility to reconstruct actor's faces in close-up video shooting.

6.2.1.3 Template initialization

The last improvement which could be carried out in the short term involves the template mesh. We currently use the surface of the first frame of the sequence to initialize the dynamic mesh. We already stated that this pose should be adapted to the morphology of the actor and contain neither inconsistency nor artifact in its surface reconstruction. To avoid this constraint, another approach could be to select another pose in the sequence (*i.e.* the most suited to be used as template) and to perform the tracking from this position. Therefore, starting from a pose at time $t \in [0, N]$, the motion estimation and the mesh deformation would be performed both in the forward direction (from t to N , like in the current method) and in the backward direction (from t to 0). Indeed, as our process makes no assumption about the motion occurring between two poses, it could work identically in a reverse direction.

6.2.2 Long-term

Beyond the short-term improvements that could be considered following the RECOVER 3D project, the evolution of multi-view reconstruction technologies could lead to several long-term innovations.

6.2.2.1 Time-evolving topology

Our pose-tracking method is based on a surface deformation which maintains a constant connectivity between the vertices during the animation. This criterion is an advantage for the use of our models in post-production software (see Section 5.4) but is a limitation when the visual hull change their morphology (large deformation of the shape) and/or topology (genus of the surface) during the sequence (see Section 5.5). The first solution is to re-initialize the complete process when the template surface is no longer appropriate to the next pose of the sequence. A new template could be constructed from this pose and the tracking process can then be re-started. This way, a complex sequence can be represented by several consecutive animations. The state-of-the-art methods proposed to handle this problem imply a resampling of the surface [90], which means that the number of vertices and the triangulation are modified. Even if these modifications are limited to the area where the surface collapse or split, it is not compatible with our temporal consistency criterion.

6.2.2.2 Evolution of the multi-viewpoint capture technology

Another possible evolution for our research is to reach real-time performances, which then could authorize the use of temporally consistent models in an online reconstruction process and broadcasting. Finally, we only considered in this thesis the reconstruction of a single actor. The larger studio available at ILOI provides a larger volume of capture (see Section 1.3.2). With the improvement of reconstruction using higher definition cameras,

it will be possible to reconstruct scenes containing several actors and even a scenery. The reconstruction of such scenes have been studied in state-of-the-art model-based approaches [97, 66]. The use of larger indoor studios that allows the shooting of complex scenes, instead of a single actor's performance, is a current evolution in multi-view reconstruction and is explored by several projects (*e.g.*, *Panoptic Studio* or *Kinovis*, see Section 2.2.4). Following the same path, the multi-view reconstruction could also be applied to wide, outdoor, environments such as sportive event shooting (see [76] for instance).

Conclusion générale

Bilan

Nous avons présenté dans ce manuscrit une nouvelle méthode permettant de générer un modèle dynamique temporellement cohérent à partir de séquences d'objets issues d'une reconstruction multi-vues par enveloppe visuelle. Dans le contexte de la création d'une chaîne de production TV, nous avons constaté que les méthodes de reconstruction à partir d'images vidéos multi-points de vues sont dépourvues de continuité temporelle et donc inadaptées pour l'animation de scènes virtuelles. Notre objectif était donc de développer une méthode de suivi de pose à partir d'un modèle de maillage dynamique, de manière à générer une animation 3D à partir des formes statiques reconstruites à des pas de temps réguliers. L'état de l'art a permis d'identifier deux familles de méthodes de reconstruction multi-vues. Les *méthodes libres* effectuent une reconstruction sans *a priori* sur le contenu de la scène. Cependant ces algorithmes, les plus répandus et les moins complexes, effectuent en fait une reconstruction statique à partir d'un ensemble d'images fixes. Dans le cadre de scènes dynamiques, une reconstruction est effectuée sur chaque *frame* des vidéos synchronisées, indépendamment du reste de la performance qui est capturée. Les résultats sont donc dépourvus de toute continuité temporelle et sont constitués de séquences de formes 3D fixes. Les méthodes *basées modèle* utilisent comme données d'entrée un modèle spécifique adapté à la forme à reconstruire (dans le cas d'acteurs, il s'agit le plus souvent de modèles de morphologie humaine articulés par un *squelette*) qui est animé selon un ensemble de contraintes extraites des vidéos multi-points de vues ou d'une reconstruction libre initiale. Cette approche produit des animations temporellement cohérentes telles que nous les recherchons. Elle impose cependant de contruire un modèle spécifique à l'acteur reconstruit en amont de la capture vidéo, ce qui rend leur mise en oeuvre complexe, peu générique et donc inadapté à notre contexte. La méthode que nous avons développée est inspirée de ces deux approches : nous cherchons à déformer un modèle par suivi de l'enveloppe visuelle, tout en utilisant une méthode de *morphing* adaptée aux mouvements libres et à différents types de morphologie. Notre approche est constituée de deux étapes principales. La première consiste à extraire un champ de déplacements, qui représente les mouvements de l'acteur, à partir de la succession de poses statiques. Cette étape est basée sur un algorithme d'appariement de voxels entre deux frames consécutives pour obtenir un ensemble de vecteurs 3D. L'étape suivante consiste à initialiser un maillage représentant l'acteur à la pose initiale de la séquence. Cette surface est ensuite déformée selon les mouvements décrits par le champ de déplacements de manière à adopter les poses successives décrites

par la reconstruction initiale. Cette déformation de maillage est basée sur des algorithmes de type *As-Rigid-As-Possible* et est ainsi adaptée aux mouvements de surfaces libres, tout en respectant une contrainte de rigidité locale qui assure la conservation d'une triangulation cohérente tout au long de l'animation. Nous avons détaillé les résultats obtenus à partir de différents jeux de données et montré l'influence des différents paramètres de nos algorithmes sur la qualité des résultats finaux. Les maillages animés obtenus grâce à notre méthode peuvent être aisément adaptés aux formats standards des logiciels de modélisation et animation 3D utilisés pour la création et le rendu de scène virtuelles. De cette manière, l'acteur reconstruit sous forme de personnage animé peut être placé dans un environnement généré par ordinateur de la même manière qu'un objet modélisé par un artiste (et non plus sous la formes de séquences où chaque pose doit être chargée lors du rendu de la *frame* correspondante). La cohérence temporelle qui caractérise notre modèle assure que la connectivité du maillage reste constante au cours de l'animation et que seules les positions des sommets soient modifiées. De cette manière, la surface du clone virtuel peut être suivie au cours de l'animation, ce qui permet de détecter les collisions avec des objets de la scène 3D. Le personnage peut ainsi interagir avec l'environnement virtuel dans lequel il est placé (immersion dans un flot de particules, accessoires virtuels ...). Les maillages animés que nous produisons permettent également d'améliorer la qualité visuelle de la reconstruction et de simplifier l'application de texture sur le modèle.

Perspectives

Les résultats de nos recherches ouvrent la voie à plusieurs travaux futurs. A court terme, une première amélioration pourrait être développée pour utiliser comme données d'entrée des reconstructions de meilleure qualité. Tout d'abord, une enveloppe visuelle plus précise pourrait être calculée en utilisant des grille de voxels de plus haute résolution. Cette amélioration nécessite cependant de mettre en place une structure de données optimisée pour contenir l'ensemble des données nécessaires à nos algorithmes (volume RGB, EDT, champs de déplacements...). Parmi les approches envisageables, des structures hiérarchiques de type *octree* pourraient être utilisées. On peut de plus noter que la plupart de ces données sont éparées (voxels de surface, vecteurs de mouvements, valeurs de l'EDT inutilisées à partir d'une certaine distance à la surface) et pourrait donc être stockées dans des grilles irrégulières. Le projet RECOVER 3D prévoit également d'améliorer la reconstruction multi-vues grâce à la mise en place de reconstruction par *stéréo-matching* à partir des blocs de caméras multiscopiques. Ainsi, l'enveloppe visuelle pourrait être *creusée* par des carte de profondeurs calculées à partir de différents points de vues. Cette méthode nécessite elle aussi des grilles de voxels de hautes résolutions. D'autres approches basées nuages de points pourraient être mises en place. Il sera dans tous les cas nécessaire d'adapter notre méthode à ces nouvelles données d'entrée. On peut également noter que l'augmentation de la résolution des volumes est également indispensable dans l'hypothèse où les caméras HD utilisées dans la version actuelle du studio RECOVER3D seraient remplacées par des caméras 4K. Cette amélioration de l'infrastructure, associées à la mise en place de la stéréoscopie, permettrait d'obtenir dès l'étape initiale de reconstruction, des volumes de grande précision, riches en détails. Ces évolutions permettraient également d'effectuer des reconstructions de visages à partir de plans rapprochés. A long terme, différentes solutions peuvent être étudiées pour apporter une solution aux limitations de notre approche. La principale de ces limitations est l'incapacité de notre outil à gérer les changements de topologies qui peuvent subvenir au cours de la séquence de reconstruction. En effet notre algorithme de

suivi de poses est basé sur la déformation d'une surface dont la triangulation (et donc la topologie) est constante au cours de l'animation. Ce critère est un avantage lors de l'utilisation des modèles (voir ci-dessus) mais est inadapté dans le cas où la topologie (ou la morphologie) de l'enveloppe visuelle est modifiée entre deux poses successives (par exemple un bras se collant au torse). Plusieurs approches décrites dans l'état de l'art proposent des solutions pour le suivi de telles formes [24, 90] où le modèle de maillage modifie sa connectivité uniquement dans les zones où apparaissent de tels changements. Enfin, nous avons jusqu'ici limité nos développements à des scènes ne contenant qu'un seul acteur. Suite à la construction d'un second studio RECOVER3D à l'ILOI, de taille supérieure au studio parisien, les dimensions accrues de la zone de capture permettent d'envisager la reconstruction de scènes contenant plusieurs acteurs, et d'éventuels éléments de décors. Cette piste de recherche semble également pertinente au vu de l'évolution de la reconstruction multi-vues, où plusieurs projets (*Panoptic Studio*, *Kinovis*...) développent des studios de grande taille et cherchent à adapter ces solutions à la capture en plein air.

References

- [1] Y.I. Abdel-Aziz, H.M. Karara, and Michael Hauck. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Photogrammetric Engineering & Remote Sensing*, 81(2):103 – 107, 2015.
- [2] Naveed Ahmed, Christian Theobalt, Petar Dobrev, Hans-Peter Seidel, and Sebastian Thrun. Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.
- [3] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.
- [4] Marc Alexa and Anders Adamson. On normals and projection operators for surfaces defined by point sets. In *Proceedings of the First Eurographics Conference on Point-Based Graphics*, SPBG’04, pages 149–155, June 2004.
- [5] Oleg Alexander, Graham Fyffe, Jay Busch, Xueming Yu, Ryosuke Ichikari, Andrew Jones, Paul Debevec, Jorge Jimenez, Etienne Danvoye, Bernardo Antionazzi, Mike Eheler, Zybnek Kysela, and Javier von der Pahlen. Digital Ira: Creating a Real-Time Photoreal Digital Actor. In *SIGGRAPH Real Time Live!*, Anaheim, CA, July 2013.
- [6] Benjamin Allain, Jean-Sébastien Franco, and Edmond Boyer. An Efficient Volumetric Framework for Shape Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, United States, June 2015.
- [7] Benjamin Allain, Jean-Sébastien Franco, Edmond Boyer, and Tony Tung. On mean pose and variability of 3D deformable models. In *ECCV 2014 - European Conference on Computer Vision*, pages 284–297, September 2014.
- [8] Jérémie Allard, Jean-Sébastien Franco, Clément Ménier, Edmond Boyer, and Bruno Raffin. The GrImage platform: A mixed reality environment for interactions. In *IEEE International Conference on Computer Vision Systems (ICVS)*, pages 46–46, January 2006.
- [9] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*, 21(3):612–619, July 2002.
- [10] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, 22(3):587–594, July 2003.
- [11] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

-
- [12] Dragomir Anguelov, Praveen Srinivasan, Hoi cheung Pang, Daphne Koller, Sebastian Thrun, and James Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Advances in Neural Information Processing Systems 17*, pages 33–40. MIT Press, 2004.
 - [13] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: Shape completion and animation of people. *ACM Transactions on Graphics - Proceedings of ACM SIGGRAPH*, 24(3):408–416, July 2005.
 - [14] Nizam Anuar and Igor Guskov. Extracting animated meshes with adaptive motion estimation. In *International Workshop on Vision, Modeling and Visualization (VMV)*, pages 63–71, November 2004.
 - [15] Romain Arcila. *Séquence de maillages : classification et méthodes de segmentation*. PhD thesis, Université de Lyon, November 2011.
 - [16] Romain Arcila, Cédric Cagniart, Franck Hétroy, Edmond Boyer, and Florent Dupont. Segmentation of temporal mesh sequences into rigidly moving components. *Graphical Models*, 75(1):10 – 22, 2013.
 - [17] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.
 - [18] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics*, 26(3), July 2007.
 - [19] John L. Barron and Neil A. Thacker. Tutorial: Computing 2D and 3D optical flow. Technical Report 2004-012, Tina Memo, 2004.
 - [20] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.
 - [21] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, February 1992.
 - [22] Eric Bittar, Aassif Benassarou, Dominique Ploton, and Laurent Lucas. Hierarchical tracking of intra-cell structures in 4d images. In *3rd International Conference on BioMedical Visualization*, pages 82–90, July 2006.
 - [23] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, pages 187–194, 1999.
 - [24] Morten Bojsen-Hansen, Hao Li, and Chris Wojtan. Tracking surfaces with evolving topology. *ACM Transactions on Graphics*, 31(4):53:1–53:10, July 2012.
 - [25] Francesco Bonarrigo, Alberto Signoroni, and Mario Botsch. Deformable registration using patch-wise shape matching. *Graphical Models*, 76(5):554–565, September 2014.
 - [26] Mario Botsch, Mark Pauly, Martin Wicke, and Markus Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3):339–347, 2007.
 - [27] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):213–230, January 2008.
 - [28] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Dynamic 2D/3D Registration. In Nicolas Holzschuch and Karol Myszkowski, editors, *Eurographics 2014 - Tutorials*. The Eurographics Association, 2014.

-
- [29] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 26–33 vol.1, October 2003.
 - [30] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Transactions on Graphics*, 27(3):99:1–99:9, August 2008.
 - [31] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07. ACM, 2007.
 - [32] Fernando Cacciola. Triangulated surface mesh simplification. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.6 edition, 2015.
 - [33] Cédric Cagniard, Edmond Boyer, and Slobodan Ilic. Probabilistic deformable surface tracking from multiple videos. In *11th European conference on Computer vision (ECCV)*, volume 6314 of *Lecture Notes in Computer Science*, pages 326–339, 2010.
 - [34] Joel Carranza, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics*, 22(3):569–577, July 2003.
 - [35] Will Chang, Hao Li, Niloy Mitra, Mark Pauly, Szymon Rusinkiewicz, and Michael Wand. Computing correspondences in geometric data sets. In *Eurographics 2011 - Tutorials*, 2011.
 - [36] Will Chang and Matthias Zwicker. Range scan registration using reduced deformable models. *Computer Graphics Forum*, 28(2):447–456, 2009.
 - [37] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing - Special issue: range image understanding*, 10(3):145–155, April 1992.
 - [38] German K.M. Cheung, Simon Baker, and Takeo Kanade. Visual hull alignment and refinement across time: a 3D reconstruction algorithm combining shape-from-silhouette with stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 375–82, June 2003.
 - [39] German K.M. Cheung, Takeo Kanade, Jean-Yves Bouguet, and Mark Holler. A real time system for robust 3D voxel reconstruction of human motions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 714–720, 2000.
 - [40] Chiun-Hong Chien and Jake K. Aggarwal. Volume/surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 36(1):100–113, November 1986.
 - [41] Laurent D. Cohen and Isaac Cohen. Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.
 - [42] Stefano Corazza, Lars Mündermann, Ajit M. Chaudhari, T. Demattio, Claudio Cobelli, and Thomas P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, 2006.
 - [43] Stefano Corazza, Lars Mündermann, Emiliano Gambaretto, Giancarlo Ferrigno, and Thomas P. Andriacchi. Markerless motion capture through visual hull, articulated icp and subject specific model generation. *International Journal of Computer Vision*, 87:156–169, March 2010.

-
- [44] Boguslaw Cyganek and J. Paul Siebert. *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, 2009.
 - [45] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics*, 27(3):98:1–98:10, August 2008.
 - [46] Edilson de Aguiar, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Reconstructing human shape and motion from multi-view video. In *Visual Media Production, 2005. CVMP 2005. The 2nd IEEE European Conference on*, pages 44–51, November 2005.
 - [47] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Markerless deformable mesh tracking for human shape and motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.
 - [48] Hervé Delingette. General object reconstruction based on simplex meshes. *International Journal of Computer Vision*, 32(2):111–146, August 1999.
 - [49] David Demirdjian. Combining geometric- and view-based approaches for articulated pose estimation. In *Computer Vision - ECCV 2004*, volume 3023 of *Lecture Notes in Computer Science*, pages 183–194. Springer Berlin Heidelberg, 2004.
 - [50] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
 - [51] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95*, pages 287–290, 1995.
 - [52] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, pages 317–324, 1999.
 - [53] François Destelle, Céline Roudet, Marc Neveu, and Albert Dipanda. Toward a real-time tracking of dense point-sampled geometry. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 381–384, September 2012.
 - [54] Harish Doraiswamy and Vijay Natarajan. Efficient algorithms for computing reeb graphs. *Computational Geometry*, 42(6-7):606–616, 2009.
 - [55] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, December 1999.
 - [56] Estelle Duveau, Simon Courtemanche, Lionel Reveret, and Edmond Boyer. Cage-based motion recovery using manifold learning. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 206–213, October 2012.
 - [57] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding - Model-based and image-based 3D scene representation for interactive visualization*, 96(3):367–392, December 2004.
 - [58] Jean-Sébastien Franco and Edmond Boyer. Exact polyhedral visual hulls. *British Machine Vision Conference*, 2003.

-
- [59] Ryo Furukawa, Ryusuke Sagawa, Hiroshi Kawasaki, Kazuhiro Sakashita, Yasushi Yagi, and Naoki Asada. One-shot entire shape acquisition method using multiple projectors and cameras. In *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*, pages 107–114, November 2010.
 - [60] Yasutaka Furukawa and Jean Ponce. Dense 3D motion capture from synchronized video streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.
 - [61] Juergen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1753, June 2009.
 - [62] Dirk Hähnel, Sebastian Thrun, and Wolfram Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI’03*, pages 915–920, 2003.
 - [63] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.
 - [64] Radu P. Horaud, Matti Niskanen, Guillaume Dewaele, and Edmond Boyer. Human motion tracking by registering an articulated surface to 3-d points and normals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):158–164, January 2009.
 - [65] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.
 - [66] Chun Hao Huang, Edmond Boyer, Nassir Navab, and Slobodan Ilic. Human Shape and Pose Tracking Using Keyframes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3446–3453. IEEE, June 2014.
 - [67] Peng Huang, Jonathan Starck, and Adrian Hilton. A study of shape similarity for temporal surface sequences of people. In *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, pages 408–418, 2007.
 - [68] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. In *Proceedings of the Symposium on Geometry Processing, SGP ’08*, pages 1449–1457, 2008.
 - [69] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST ’11*, pages 559–568, 2011.
 - [70] Hanbyul Joo, Hyun Soo Park, and Yaser Sheikh. Map visibility estimation for large-scale dynamic 3D reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1122–1129, June 2014.
 - [71] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3), July 2007.
 - [72] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, July 2005.

-
- [73] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, January 1997.
 - [74] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
 - [75] Roland Kehl and Luc Van Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2-3):190–209, 2006.
 - [76] Joe Kilner, Jean-Yves Guillemaut, and Adrian Hilton. 3D action matching with key-pose detection. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1–8, October 2009.
 - [77] Joe Kilner, Jonathan Starck, Adrian Hilton, and Oliver Grau. Dual-mode deformable models for free-viewpoint video of sports events. In *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, pages 177–184, 2007.
 - [78] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 105–114, 1998.
 - [79] Koji Komatsu. Human skin model capable of natural shape variation. *The Visual Computer*, 3(5):265–271, March 1988.
 - [80] Denis Kravtsov, Oleg Fryazinov, Valery Adzhiev, Alexander Pasko, and Peter Conninos. Controlled metamorphosis between skeleton-driven animated polyhedral meshes of arbitrary topologies. *Computer Graphics Forum*, 33(1):64–72, 2014.
 - [81] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000.
 - [82] Jacques-Olivier Lachaud and Annick Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
 - [83] Aldo Laurentini. Visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.
 - [84] Svetlana Lazebnik, Edmond Boyer, and Jean Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume I, pages 151–161, December 2001.
 - [85] Antoine Letouzey and Edmond Boyer. Progressive shape models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 190–197, June 2012.
 - [86] Antoine Letouzey, Benjamin Petit, and Edmond Boyer. Scene flow from depth and color images. In *BMVC 2011 - British Machine Vision Conference*, Proceedings of the British Machine Vision Conference, pages 46:1–11, August 2011.
 - [87] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of the 29th ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, July 2002.
 - [88] John P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. Practice and theory of blendshape facial models. In Sylvain Lefebvre and Michela Spagnuolo, editors, *EG 2014 - STARs*, pages 199–218, 2014.

-
- [89] John P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 165–172, 2000.
 - [90] Hao Li, Bart Adams, Leonidas J. Guibas, and Mark Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics*, 28(5):175:1–175:10, December 2009.
 - [91] Hao Li, Linjie Luo, Daniel Vlasic, Pieter Peers, Jovan Popović, Mark Pauly, and Szymon Rusinkiewicz. Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics*, 31(1):2:1–2:11, February 2012.
 - [92] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1421–1430, 2008.
 - [93] Reoxiang Li, Bing Zeng, and M. L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Trans. Cir. and Sys. for Video Technol.*, 4(4):438–442, August 1994.
 - [94] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization '98. Proceedings*, pages 279–286, October 1998.
 - [95] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Transactions on Graphics*, 27(3), August 2008.
 - [96] Yebin Liu, Qionghai Dai, and Wenli Xu. A point-cloud-based multiview stereo algorithm for free-viewpoint video. *Visualization and Computer Graphics, IEEE Transactions on*, 16(3):407–418, May 2010.
 - [97] Yebin Liu, Carsten Stoll, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Markerless motion capture of interacting characters using multi-view image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1249–1256, June 2011.
 - [98] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, August 1987.
 - [99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2 of *ICCV '99*, pages 1150–1157, 1999.
 - [100] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.
 - [101] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.
 - [102] Weilan Luo, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cooperative estimation of human motion and surfaces using multiview videos. *Computer Vision and Image Understanding*, 117(11):1560–1574, November 2013.
 - [103] Zhiping Luo, Remco C. Veltkamp, and Arjan Egges. *Advances in Visual Computing: 11th International Symposium, ISVC 2015, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part I*, chapter As-Rigid-As-Possible Character Deformation Using Point Handles, pages 57–70. Springer International Publishing, 2015.

-
- [104] Marcus A. Magnor, Oliver Grau, Olga Sorkine-Hornung, and Christian Theobalt, editors. *Digital Representations of the Real World: How to Capture, Model, and Render Visual Reality*. A K Peters/CRC Press, 2015.
 - [105] Stephen R. Marschner, Brian Guenter, and Sashi Raghupathy. Modeling and rendering for realistic facial animation. In *Rendering Techniques 2000*, Eurographics, pages 231–242. Springer Vienna, 2000.
 - [106] Worthy N. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.
 - [107] Takashi Matsuyama, Shohei Nobuhara, Takeshi Takai, and Tony Tung. *3D Video and Its Applications*. Springer Publishing Company, Incorporated, 2012.
 - [108] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *In Proceedings of Twelfth Eurographics Workshop on Rendering*, pages 115–125, 2001.
 - [109] Ivana Mikić, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, 2003.
 - [110] Niloy J. Mitra, Simon Flory, Maks Ovsjanikov, Natasha Gelfand, Leonidas Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Symposium on Geometry Processing*, pages 173–182, 2007.
 - [111] Thomas B. Moeslund, Adrian Hilton, Volker Krüger, and Leonid Sigal. *Visual Analysis of Humans: Looking at People*. Springer Publishing Company, Incorporated, 2013.
 - [112] Tomoyuki Mukasa, Shohei Nobuhara, Atsuto Maki, and Takashi Matsuyama. Finding articulated body in time-series volume data. In *Articulated Motion and Deformable Objects*, volume 4069 of *Lecture Notes in Computer Science*, pages 395–404. Springer Berlin Heidelberg, 2006.
 - [113] Lars Mündermann, Stefano Corazza, and Thomas P. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, June 2007.
 - [114] Jan Neumann and Yiannis Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 103–108, 2001.
 - [115] Yao Nie and Kai-Kuang Ma. Adaptive rood pattern search for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 11(12):1442–1449, December 2002.
 - [116] Wolfgang Niem and Ralf Buschmann. Automatic modelling of 3D natural objects from multiple views. In Yakup Paker and Sylvia Wilbur, editors, *Image Processing for Broadcast and Video Production*, Workshops in Computing, pages 181–193. Springer London, 1995.
 - [117] Shohei Nobuhara and Takashi Matsuyama. Heterogeneous deformation model for 3D shape and motion recovery from multi-viewpoint images. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, pages 566–573, September 2004.

-
- [118] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, November 1988.
 - [119] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Transactions on Graphics*, 25(3):881–889, July 2006.
 - [120] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, VIS*, pages 163–170, November 2002.
 - [121] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus Gross, and Leonidas J. Guibas. Example-based 3D scan completion. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05. Eurographics Association, 2005.
 - [122] Yuri Pekelný and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum*, 27(2):399–408, 2008.
 - [123] Wei Peng, Detang Lu, Tao Huang, and Rongwang Yin. As-rigid-as-possible mesh deformation and its application in hexahedral mesh generation. *Advances in Engineering Software*, 65:158 – 167, 2013.
 - [124] Benjamin Petit, Antoine Letouzey, Edmond Boyer, and Jean-Sébastien Franco. Surface flow from visual cues. In *International Workshop on Vision, Modeling and Visualization (VMV)*, pages 1–8, October 2011.
 - [125] Ralf Plankers and Pascal Fua. Articulated soft objects for multiview shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, 2003.
 - [126] Tiberiu Popa, Ian South-Dickinson, Derek Bradley, Alla Sheffer, and Wolfgang Heidrich. Globally consistent space-time reconstruction. *Computer Graphics Forum - Proceedings of Eurographics Symposium on Geometry Processing*, 29(5), 2010.
 - [127] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, October 2007.
 - [128] Emil Praun and Hugues Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, July 2003.
 - [129] Rémi Ronfard and Gabriel Taubin. *Image and Geometry Processing for 3-D Cinematography*, volume 5 of *Geometry and Computing*. Springer Berlin Heidelberg, 2010.
 - [130] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, July 2002.
 - [131] Toyofumi Saito and Jun-Ichiro Toriwaki. New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, 27(11):1551–1565, 1994.
 - [132] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM Transactions on Graphics*, 22(3):578–586, July 2003.
 - [133] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.
 - [134] Oliver Schreer, Peter Kauff, and Thomas Sikora. *3D Videocommunication: Algorithms, Concepts and Real-time Systems in Human Centred Communication*. John Wiley & Sons, 2005.

-
- [135] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528, 2006.
 - [136] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.
 - [137] Greg Slabaugh, Bruce Culbertson, Tom Malzbender, and Ron Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *Proceedings of the 2001 Eurographics Conference on Volume Graphics*, VG’01, pages 81–101, 2001.
 - [138] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP*, pages 109–116, 2007.
 - [139] Jonathan Starck and Adrian Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, October 2007.
 - [140] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.
 - [141] Jonathan Starck, Atsuto Maki, Shohei Nobuhara, Adrian Hilton, and Takashi Matsuyama. The multiple-camera 3-D production studio. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(6):856–869, June 2009.
 - [142] Dvir Steiner and Anath Fischer. Cutting 3D freeform objects with genus- n into single boundary surfaces using topological graphs. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, SMA ’02, pages 336–343, 2002.
 - [143] Steve Sullivan and Jean Ponce. Automatic model construction, pose estimation, and object recognition from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1091–1096, 1998.
 - [144] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, August 2004.
 - [145] Jochen Süßmuth, Marco Winter, and Günther Greiner. Reconstructing animated meshes from time-varying point clouds. In *Proceedings of the Symposium on Geometry Processing*, SGP ’08, pages 1469–1476, 2008.
 - [146] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, July 1993.
 - [147] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
 - [148] Ryuichi Tadano, Toshihiko Yamasaki, and Kiyoharu Aizawa. Fast and robust motion tracking for time-varying mesh featuring reeb-graph-based skeleton fitting and its application to motion retrieval. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 2010–2013, July 2007.
 - [149] Takeshi Takai, Shohei Nobuhara, Hiromasa Yoshimoto, and Takashi Matsuyama. 3D video technologies: Capturing high fidelity full 3D shape, motion, and texture. In *International Workshop on Mixed Reality Technology for Filmmaking (in cooperation with ISMAR 2006)*, 2006.
 - [150] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’95, pages 351–358, 1995.

-
- [151] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(4):413–424, July 1986.
 - [152] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1988.
 - [153] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, Martin Bokeloh, Jens Kerber, and Hans-Peter Seidel. Animation cartography - intrinsic reconstruction of shape and motion. *ACM Transactions on Graphics*, 31(2):12:1–12:15, April 2012.
 - [154] Art Tevs, Martin Bokeloh, Michael Wand, Andreas Schilling, and Hans-Peter Seidel. Isometric registration of ambiguous and partial data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1185–1192, June 2009.
 - [155] Arasanathan Thayananathan, Ramanan Navaratnam, Björn Stenger, Philip H. S. Torr, and Roberto Cipolla. Multivariate relevance vector machines for tracking. In *Computer Vision - ECCV 2006*, volume 3953 of *Lecture Notes in Computer Science*, pages 124–138. Springer Berlin Heidelberg, 2006.
 - [156] Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *Visualization and Computer Graphics, IEEE Transactions on*, 13(4):663–674, July 2007.
 - [157] Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. CageR: Cage-based reverse engineering of animated 3D shapes. *Computer Graphics Forum*, 31(8):2303–2316, 2012.
 - [158] Toni Tung and Takashi Matsuyama. Dynamic surface matching by geodesic mapping for 3D animation transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1402–1409, June 2010.
 - [159] Tony Tung and Francis Schmitt. The augmented multiresolution Reeb graph approach for content-based retrieval of 3D shapes. *International Journal of Shape Modeling*, 11(01):91–120, 2005.
 - [160] Norimichi Ukita, Michiro Hirai, and Masatsugu Kidode. Complex volume and pose tracking with probabilistic dynamical models and visual hull constraints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1405–1412, September 2009.
 - [161] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Eurographics 2010 - State of the Art Reports*, 2010.
 - [162] Kiran Varanasi, Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Temporal surface tracking using mesh evolution. In *10th European Conference on Computer Vision (ECCV)*, volume 5303 of *Lecture Notes in Computer Science*, pages 30–43, 2008.
 - [163] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 722–729, 1999.
 - [164] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):97:1–97:09, August 2008.

-
- [165] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics*, 28(5):174:1–174:11, December 2009.
 - [166] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D scene flow estimation with a rigid motion prior. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, ICCV '11, pages 1291–1298, 2011.
 - [167] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3D scanner data. *ACM Transactions on Graphics*, 28(2):15:1–15:15, May 2009.
 - [168] Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 49–58, 2007.
 - [169] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, SCA*, pages 129–138, 2002.
 - [170] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3D motion understanding. *International Journal of Computer Vision*, 95(1):29–51, October 2011.
 - [171] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.
 - [172] Thibaut Weise, Bastian Leibe, and Luc Van Gool. Fast 3D scanning with automatic motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.
 - [173] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu P. Horaud. Surface feature detection and description with applications to mesh matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009.
 - [174] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High resolution capture for modeling and animation. *ACM Transactions on Graphics*, 23(3):548–558, August 2004.
 - [175] Zhengyou Zhang. Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4–10, February 2012.
 - [176] Qian Zheng, Andrei Sharf, Andrea Tagliasacchi, Baoquan Chen, Hao Zhang, Alla Sheffer, and Daniel Cohen-Or. Consensus Skeleton for Non-rigid Space-time Registration. *Computer Graphics Forum*, 29(2):635–644, 2010.
 - [177] Kun Zhou, Hujun Bao, and Jiaoying Shi. 3D surface filtering using spherical harmonics. *Computer-Aided Design*, 36(4):363–375, 2004.
 - [178] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 9(2):287–290, February 2000.

Publications and communications

Journals

Ludovic Blache, Céline Loscos and Laurent Lucas. **Robust motion flow for mesh tracking of freely moving actors**. In *The Visual Computer*, Springer Berlin Heidelberg, 32(2):205-216, 2015.

Ludovic Blache, Céline Loscos, Laurent Lucas, and Mathieu Desbrun. **Reconstruction 4D de maillages d'acteurs par suivi pseudo-rigide**. In *Revue Electronique Franco-phone d'Informatique Graphique (REFIG)*, 9(1):11-21, 2015.

International communications

Ludovic Blache, Mathieu Desbrun, Céline Loscos, and Laurent Lucas. **Time-varying surface reconstruction of an actor's performance**. In *Advances in Visual Computing* Volume 9474 of the series *Lecture Notes in Computer Science*, Springer International Publishing, *International Symposium on Visual Computing (ISVC)*, Las Vegas, Nevada, USA, December 2015.

Ludovic Blache, Mathieu Desbrun, Céline Loscos, and Laurent Lucas. **4D mesh reconstruction from time-varying voxelized geometry through ARAP tracking**. In *Eurographics (EG)*, pages 9-12, Zurich, Suisse, May 2015.

Ludovic Blache, Céline Loscos, Olivier Nocent, and Laurent Lucas. **3D volume matching for mesh animation of moving actors**. In *Eurographics Workshop on 3D Object Retrieval (3DOR)*, pages 69-76, Strasbourg, France, April 2014. Eurographics.

Laurent Lucas, Philippe Souchet, Muhannad Ismael, Olivier Nocent, Cédric Niquin, Céline Loscos, Ludovic Blache, Stéphanie Prévost, and Yannick Remion. **RECOVER3D : A hybrid multi-view system for 4D reconstruction of moving actors**. In *4th International Conference and Exhibition on 3D Body Scanning Technologies (3DBST)*, pages 219-230, Long Beach, CA, USA, November 2013. Hometrica Consulting.

Book chapters

Ludovic Blache, Muhannad Ismael, and Philippe Souchet. **3D scene reconstruction and structuring**. In *3D Video : From Capture to Diffusion*, pages 157-171. Wiley and Sons, 2013.

Ludovic Blache, Muhannad Ismael, and Philippe Souchet. **Reconstruction et structuration 3D de scènes**. In *Vidéo 3D : Capture, traitement et diffusion*, pages 165-180. Hermès/Lavoisier, 2013.

National communications

Ludovic Blache, Céline Loscos, Laurent Lucas, and Mathieu Desbrun. **Reconstruction 4D de maillages d'acteurs par suivi pseudo-rigide**. In *Journées de l'Association Française d'Informatique Graphique (AFIG)*, pages 101-109, Reims, France, November 2014.

Laurent Lucas, Philippe Souchet, Muhannad Ismael, Raïssel Ramirez Orozco, Cédric Niquin, Céline Loscos, Ludovic Blache, Stéphanie Prévost, and Yannick Remion. **RECOVER3D : système multi-vues hybride pour la reconstruction 4D de performances d'acteur(s)**. In *Journées de l'Association Française d'Informatique Graphique (AFIG)*, pages 187-188, Reims, France, November 2014.

Ludovic Blache, Olivier Nocent, Laurent Lucas, and Céline Loscos. **Extraction de mouvements à partir de séquences de volumes reconstruits**. In *Journées du Groupe de Travail en Modélisation Géométrique (GTMG)*, pages 45-51, Marseille, France, March 2013.

Seminars

Ludovic Blache. **Représentation dynamique de modèles d'acteurs issus de reconstructions multi-vues**. *Réunion GDR ISIS - Vision et modélisation 3D d'environnements dynamiques*, Télécom ParisTech, Paris, June 2016.

Ludovic Blache. **Animation de maillages dynamiques à partir de reconstructions d'acteurs**. *Journées Informatique et Géométrie (JIG)*, ESIEE Paris, Marne-la-Vallée, October 2015.

Ludovic Blache. **Reconstruction 4D de maillages d'acteurs par suivi pseudo-rigide**. *Séminaire CReSTIC - Journée des doctorants*, Reims, France, May 2015.

Ludovic Blache. **3D Volume Matching for Mesh Animation of Moving Actors**. *TITANE seminars*, INRIA Sophia-Antipolis, France, June 2014.

Muhannad Ismael, Ludovic Blache. **RECOVER3D**. *Journée scientifique Centre Image*, Reims, France, March 2014.

Appendix B

Algorithms

B.1 Euclidean Distance Transform

The EDT algorithm consists of n one-dimensional local operations performed serially, each of which corresponds to the direction of each coordinate axis. It does not use vector propagation, nor fixed template strategies and still always gives exact Euclidean distance stored as an array of squared distance values.

Data: B : binary volume of size $L \times M \times N$
Result: S : Euclidean Distance Transform grid
Derive from $B \rightarrow G = \{g_{ijk}\}$ with $g_{ijk} = \min_x \{(i-x)^2; b_{xjk} = 0, 1 \leq x \leq L\}$;
Derive from $G \rightarrow H = \{h_{ijk}\}$ with $h_{ijk} = \min_y \{g_{iyk} + (j-y)^2, 1 \leq y \leq M\}$;
Derive from $H \rightarrow S = \{s_{ijk}\}$ with $s_{ijk} = \min_z \{h_{ijz} + (k-z)^2, 1 \leq z \leq N\}$;
return S ;

Algorithm 4: Euclidean Distance Transform computation

We consider a binary image $B = \{b_{i,j,k}\}$ of size $L \times M \times N$, where $b_{i,j,k}$ is the value of the voxel at coordinates (i, j, k) . We note $d((i, j, k), (p, q, r))$ the Euclidean distance between $b_{i,j,k}$ and $b_{p,q,r}$. The Euclidean Distance Transform of the image is noted $S = \{s_{i,j,k}\}$. We define $s_{i,j,k}$ as the minimum distance between $b_{i,j,k}$ and the closest point of the object (*i.e.* voxel with value 0):

$$s_{i,j,k} = \min_{(p,q,r)} \{d((i, j, k), (p, q, r))^2 \quad / \quad b_{p,q,r} = 0\}$$

$$= \min_{(p,q,r)} \{(i-p)^2 + (j-q)^2 + (k-r)^2 \quad / \quad b_{p,q,r} = 0\}.$$

with $1 \leq p \leq L, 1 \leq q \leq M, 1 \leq r \leq N$.

The algorithm is based on forward and backward scan on each spatial axis (see pseudo-code Algorithm 4). Figure B.1 illustrates the first scan along X axis: the forward scan writes in each void voxel v_{ijk} of the line i the squared distance to the closest zero voxel on its left. The same computation is operated in the backward step, whereas this time only the minimum values between forward and backward scans are stored. This step is repeated for each row on the grid. It is only applied on voxels (i, j, k) which belong to the complement of the object (external voxels only). The next scans are applied on the complete domain. The

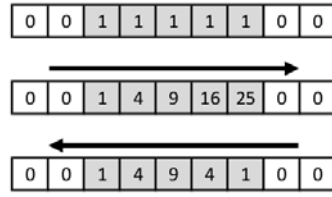


FIGURE B.1 – First step of EDT computation. Top to bottom: a single row of the initial binary grid, result after forward path along X axis (from left to right) and final result after backward path.

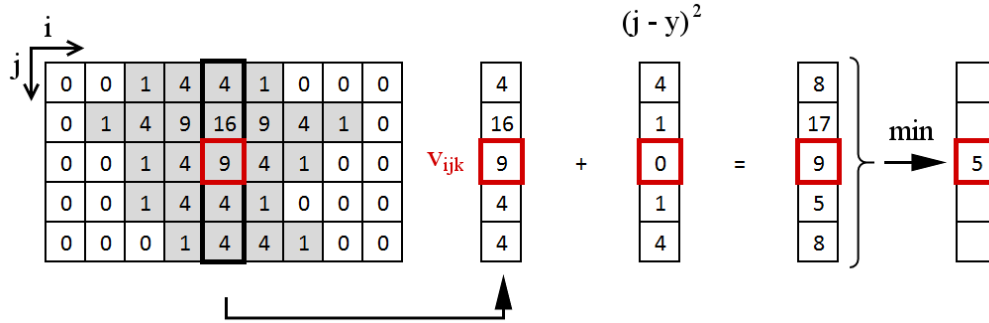


FIGURE B.2 – Second step of EDT computation. After the X axis scan (left), the new value of v_{ijk} (squared in red) is computed by a new scan along the Y axis. We extract the column i and sum each of its contained values with the result of $(j - y)^2$. The minimum value in the resulting columns is the new value of v_{ijk} .

second scan, along the Y axis, is detailed in Figure B.2: for each voxel v_{ijk} , we consider the complete column i . We then compute for each voxel v_{iyyk} of this column the value $(j - y)^2$ (i.e., the squared distance to v_{ijk}). We then compute for each voxel v_{iyyk} the sum of this distance and its original value. The minimum value of the resulting column is attributed to v_{ijk} . The same processing is applied for the Z axis scan in case of 3D grids.

This algorithm takes as input a binary volume where voxels are labelled 0 for internal points and 1 for external points. The output is a squared Euclidean distance associated to each external voxel. To compute a combined EDT, the function is applied a second time, on the inverse mask of the initial volume. The two EDT are added to get the final combination of internal and external unsigned distance transform (see Figure B.3).

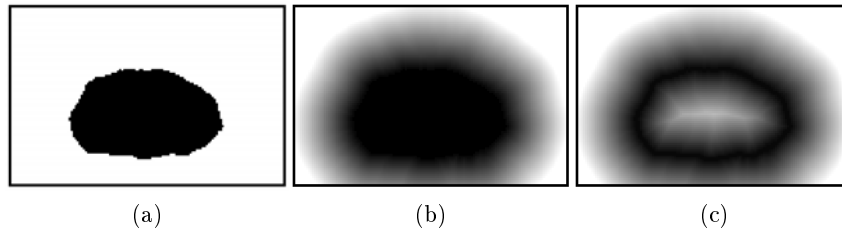


FIGURE B.3 – EDT computation: (a) original binary volume, (b) external EDT, (c) combined EDT (internal + external).

B.2 Poisson disk sampling

The principle of the Poisson disk sampling method is to generate several new points around each point already sampled. Each of this new point is checked to know whether it contains any other point in a fixed radius around itself. If no other point is identified, the new point is conserved as a sampled point (see Figure B.4). The algorithm is iteratively repeated on the growing set of sampled points until the whole space to sample is covered. Several implementations use a grid that discretizes the space to perform fast lookups of points [31].

In our case, we wish to sample the points of a discretized surface (*i.e.*, our triangulated mesh). Instead of creating new points in a continuous space, we look for samples in the set of surface's vertices. We also use a 3D grid, similar to the voxel grid of the volume, as a discretization of the space. Each square is associated with the mesh's vertex it contains. If a square contains more than one vertices, we choose one of them randomly. In other cases, using another space's discretization where the squares could be sensitively larger than the triangles and thus often contains several vertices, we could choose, for instance, the vertex which position is closer to the barycenter of these vertices. This 3D grid, containing the data, is the *input grid*. Another similar grid is created to contain the sampled point and is named *output grid*. This output grid is void at the beginning of the algorithm. A void list is also initialized and named *processing list*.

The following steps are iteratively repeated until the sampling of the complete triangulated surface (see Algorithm 5 for pseudo-code details):

1. The algorithm starts with an arbitrary point (it can be a randomly selected vertex) from the input grid. This point is both added to the output grid and the processing list.
2. A point is selected and removed from the processing list.
3. We then randomly select a set of vertices located on a sphere of fixed radius around this point. To find a random point P_2 around a fixed point P_1 , we compute the following 3D coordinates:

$$\begin{aligned} P_{2_x} &= P_{1_x} + r \cdot \cos(\alpha_1) \sin(\alpha_2) \\ P_{2_y} &= P_{1_y} + r \cdot \sin(\alpha_1) \sin(\alpha_2) \\ P_{2_z} &= P_{1_z} + r \cdot \cos(\alpha_2) \end{aligned} \tag{B.1}$$

The value r is a radius which value is randomly selected between a minimum radius $minr$ (*i.e.*, the distance criterion of the Poisson disk algorithm) and a maximum radius $maxr$ (in our implementation, we used $maxr = 2minr$). The value of two angles α_1 and α_2 are also randomly selected.

4. For each of these randomly selected vertices, we scan their minimum distance (*mindist*) neighborhood in the output grid to check if it contains any point. If it does not, the new point is added to the process list and the output grid.
5. Return to step 2 until the processign list is empty.

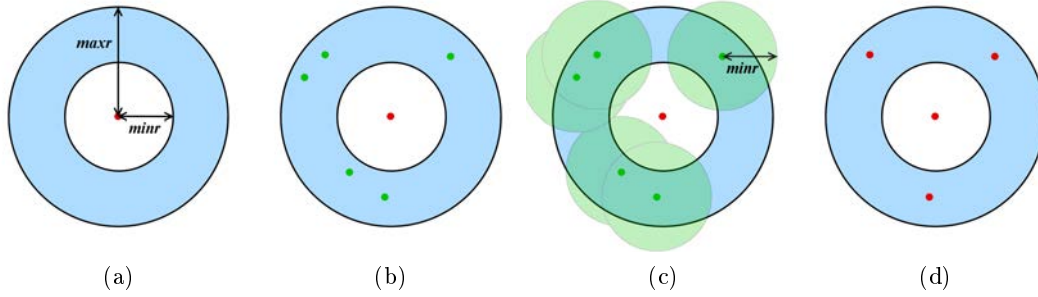


FIGURE B.4 – **Example of Poisson Disk sampling in a 2D space:** Around an initial point (red dot), a set of random points are positionned (green dots). These new points are conserved if they do not have any other point in their neighborhood.

Data: input 3D grid where each square is associated to a vertex (inputGrid), radius of the disk (radius), constant number of random point creation (cstNb)
Result: list of sampled points
 Create processList, resultList;
 firstPoint \leftarrow get random point in inputGrid;
 add firstPoint to processList;
 add firstPoint to outputGrid;
 add firstPoint to resultList;
while processList is not empty **do**
 processingPoint \leftarrow pop a random point from processList;
 compteur \leftarrow 0;
 while compteur < cstNb **do**
 newPoint \leftarrow generate random point around processingPoint between radius and radius, see eqB.1;
 if no point is found in outputGrid the radius neighborhood around newPoint **then**
 add newPoint to processList;
 add newPoint to outputGrid;
 add firstPoint to resultList;
 cpt ++;
 end
 end
end
return resultList;

Algorithm 5: Poisson disk vertex sampling

Mathematical details

C.1 Discretization of the Gaussian function

A Gaussian function is a *normal distribution* which probability density is given by the following expression:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (\text{C.1})$$

It can be noted in the form:

$$X \sim \mathcal{N}(\mu, \sigma)$$

with μ being the expectation of the distribution and σ the standard deviation. A Gaussian function centered on zero can be represented as a *probability density function of a normally distributed random variable*, *i.e.*, a normal distribution with $\mu = 0$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \frac{x^2}{\sigma^2}} \quad (\text{C.2})$$

$$X \sim \mathcal{N}(0, \sigma)$$

The value of σ can be deduced, knowing two fixed values: the half-width of the filter, noted m , and the percentage p of the Gaussian's integral we want to include in the window. In our implementation, we fixed $p = 95\%$. The value of m depends on the filter's size chosen by the user. Let us define:

$$T = \frac{X}{\sigma}$$

$$T \sim \mathcal{N}(0, 1)$$

T is now a *reduced centered random variable*, *i.e.*, a normal distribution with $\mu = 0$ and $\sigma = 1$. The integral of this function between $-\infty$ and x is given by the following expression:

$$F(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du \quad (\text{C.3})$$

with $u = \frac{x}{\sigma}$. The value of p can therefore be described by the difference between $F(\frac{m}{\sigma})$ and $F(-\frac{m}{\sigma})$:

$$F(\frac{m}{\sigma}) - F(-\frac{m}{\sigma}) = 0.95$$

which becomes:

$$2F(\frac{m}{\sigma}) = 1.95$$

$$F(\frac{m}{\sigma}) = 0.975$$

Following the table given in appendix C.2, if $F(x) = 0.975$, then $x = 1.96$. Here, $x = \frac{m}{\sigma}$, so we can compute the value of σ :

$$\sigma = \frac{m}{1.96}$$

[illegible]

C.3 3D optical flows

These algorithms are made as a 3D equivalent of the usual 2D optical flow methods, applied on a 3D grid where each voxel is associated with a single value, such as the intensity in a grey level bitmap. These implementations are based on the computations proposed in [19].

According to the **Lucas-Kanade** assumption that the velocity $V(u, v, w)$ can be considered as constant in a neighborhood n around the point \mathbf{p} , the optical flow equation (see Chapter 3, equation 3.2) can be written with the following system:

$$\begin{aligned} I_x(q_1) V_u + I_y(q_1) V_v + I_z(q_1) V_w &= -I_t(q_1) \\ I_x(q_2) V_u + I_y(q_2) V_v + I_z(q_2) V_w &= -I_t(q_2) \\ \vdots & \\ I_x(q_n) V_u + I_y(q_n) V_v + I_z(q_n) V_w &= -I_t(q_n) \end{aligned}$$

Where each point q_i belongs to the n -size neighborhood around \mathbf{p} , $i \in [0; n]$. The partial derivatives of the image I along the spatial axis x , y , and z at time t and at the point q_i are noted $I_x(q_i)$, $I_y(q_i)$, $I_z(q_i)$, and $I_t(q_i)$, respectively. $V(u, v, w)$ is the local velocity vector which defines the displacement of q_i between two images. This system can be written under a matrix form:

$$\mathbf{v}WA = WB$$

that can be written:

$$A^T \mathbf{v}WA = A^T WB$$

that becomes:

$$\mathbf{v} = (A^T WA)^{-1} A^T WB \quad (\text{C.4})$$

with the following terms:

$$\mathbf{v} = \begin{bmatrix} V_u \\ V_v \\ V_w \end{bmatrix}, \quad A = \begin{bmatrix} I_x(q_1) & I_y(q_1) & I_z(q_1) \\ I_x(q_2) & I_y(q_2) & I_z(q_2) \\ \vdots & \vdots & \vdots \\ I_x(q_n) & I_y(q_n) & I_z(q_n) \end{bmatrix}, \quad B = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

Where \mathbf{v} is the resulting 3D vector at \mathbf{p} . The derivatives $I_x(q_i)$, $I_y(q_i)$, and $I_z(q_i)$ define the spatial gradient at position q_i and $I_t(q_i)$ is the temporal derivative. W is a $n \times n$ diagonal matrix containing Gaussian weighting factors $W_{ii} = w_i$. Thus, the equation C.4 can be written with the following terms:

$$A^T WA = \begin{bmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_x(q_i) I_z(q_i) \\ \sum_i w_i I_y(q_i) I_x(q_i) & \sum_i w_i I_y(q_i)^2 & \sum_i w_i I_y(q_i) I_z(q_i) \\ \sum_i w_i I_z(q_i) I_x(q_i) & \sum_i w_i I_z(q_i) I_y(q_i) & \sum_i w_i I_z(q_i)^2 \end{bmatrix}$$

$$A^T W B = \begin{bmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \\ -\sum_i w_i I_z(q_i) I_t(q_i) \end{bmatrix}$$

The **Horn-Schunck** method is based on the minimization of this equation:

$$\begin{aligned} \sum (I_x V_u + I_y V_v + I_z V_w + I_t) + \alpha^2 & \left[\left(\frac{\partial V_u}{\partial x} \right)^2 + \left(\frac{\partial V_u}{\partial y} \right)^2 + \left(\frac{\partial V_u}{\partial z} \right)^2 \right. \\ & \left. + \left(\frac{\partial V_v}{\partial x} \right)^2 + \left(\frac{\partial V_v}{\partial y} \right)^2 + \left(\frac{\partial V_v}{\partial z} \right)^2 + \left(\frac{\partial V_w}{\partial x} \right)^2 + \left(\frac{\partial V_w}{\partial y} \right)^2 + \left(\frac{\partial V_w}{\partial z} \right)^2 \right] \quad (\text{C.5}) \end{aligned}$$

By using the following Gauss-Seidel iterative equations:

$$\begin{aligned} V_u^{k+1} &= \bar{V}_u^k - \frac{I_x [I_x \bar{V}_u^k + I_y \bar{V}_v^k + I_z \bar{V}_w^k + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)} \\ V_v^{k+1} &= \bar{V}_v^k - \frac{I_y [I_x \bar{V}_u^k + I_y \bar{V}_v^k + I_z \bar{V}_w^k + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)} \\ V_w^{k+1} &= \bar{V}_w^k - \frac{I_z [I_x \bar{V}_u^k + I_y \bar{V}_v^k + I_z \bar{V}_w^k + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)} \end{aligned}$$

\bar{V}^k is the average of the vectors resulting of the previous iteration contained in a fixed neighborhood (5⁵). The initial velocities V_u^0 , V_v^0 and V_w^0 are usually set to zero, but it can also be initialized with another vector field, for example the result of another motion flow computation. In our case we used the result of our implementation of Lucas-Kanade, and we also kept the possibility to start with null values. We apply 50 iterations with an α value set to 1.

C.4 Rigid motion computation

This computation follows the process described By O. Sorkine [138]²⁴. The goal is to find the translation t and the rotation R that minimize:

$$\sum_{i=1}^n w_i |(R\mathbf{p}_i + \mathbf{t}) - \mathbf{q}_i|^2$$

where \mathbf{p}_i and \mathbf{q}_i , $i \in [0; n]$, belongs to two set of n points. We seek for the rigid transformation (R, t) that aligns the two sets in the least squares sense. The algorithm is:

1. Compute the weighted centroids of both point sets:

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^n w_i \mathbf{p}_i}{\sum_{i=1}^n w_i}, \quad \bar{\mathbf{q}} = \frac{\sum_{i=1}^n w_i \mathbf{q}_i}{\sum_{i=1}^n w_i}$$

2. Compute the centered vectors:

$$x_i = \mathbf{p}_i - \bar{\mathbf{p}}, \quad y_i = \mathbf{q}_i - \bar{\mathbf{q}}, \quad i = 1, 2, \dots, n$$

3. Compute the $d \times d$ covariance matrix S :

$$S = XWY^T$$

Where X and Y are the $d \times n$ matrices that have x_i and y_i as their columns, respectively, and $W = \text{diag}(w_1, w_2, \dots, w_n)$.

4. Compute the *singular value decomposition* (SVD) $S = U\Sigma V^T$. The rotation is then obtained by the following expression:

$$R = V \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \det(VU^T) \end{bmatrix} U^T$$

5. Compute the optimal translation as:

$$\mathbf{t} = \bar{\mathbf{q}} - R\bar{\mathbf{p}}$$

24. <http://igl.ethz.ch/projects/ARAP/index.php>

C.5 Laplacian matrix

The Laplacian matrix of a triangulated mesh M can be computed with uniform weights or cotangent weights [52]

With uniform weights, the Laplacian matrix L is computed with the following expression:

$$L = I - D^{-1} \cdot A$$

with D being the *degree matrix* and A the *adjacency matrix*.

The degree matrix is a diagonal matrix computed with the following expression:

$$D(i, i) = |N(i)|$$

where $N(i)$ is the immediate neighborhood of the vertex v_i . $D(i, i)$ is therefore the number of direct neighboring vertices or the number of incident edges (named *degree* or *valence*).

The adjacency matrix is computed by:

$$\begin{cases} A(i, j) = 1 & \text{if } j \in N(i) \\ A(i, j) = 0 & \text{otherwise} \end{cases}$$

With cotangent weights, the Laplacian matrix L is computed with:

$$L = D^{-1} \cdot A$$

With the degree matrix computed as follows:

$$D(i, i) = \sum_{t \in S(i)} \frac{Area(t)}{3}$$

where $Area(t)$ is the area of the triangle t and $S(i)$ is the set of incident triangles to vertex v_i .

The adjacency matrix is computed by:

$$A(i, j) = \frac{\cotan(\beta_{(i,j)}) + \cotan(\beta'_{(i,j)})}{2}$$

where $\beta_{(i,j)}$ and $\beta'_{(i,j)}$ are the opposite angles to the edge (i, j) in its two adjacent triangles.

C.6 Numerical integration

A solid object Y at time t can be defined as a combination of a position $p(t)$ and a velocity $v(t)$:

$$Y(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$$

When this object is submitted to a force f , it becomes:

$$\dot{Y}(t) = F(t) = \begin{bmatrix} v(t) \\ \frac{1}{m}f(t) \end{bmatrix}$$

with m being the mass of the object.

An approximation to the solution of this differential equation can be obtained using a numerical method. The most basic explicit method for numerical integration is the Euler (first-order explicit) method. It is a iterative approach where a timestep h is introduced.

$$Y(t+h) = Y(t) + hF(t)$$

$$Y(t+h) = \begin{bmatrix} p(t+h) = p(t) + hv(t) \\ v(t+h) = v(t) + \frac{h}{m}f(t) \end{bmatrix}$$

The values of $p(t+h)$ and $v(t+h)$ are then used as new values of $p(t)$ and $v(t)$, respectively, for the next iteration. By repeating this operation, an estimation of $\dot{Y}(t)$ can be obtained.

Among the other integration approaches, the fourth order Runge-Kutta (or $RK4$) method uses four increments defined as follow:

$$\begin{aligned} k_1 &= F(t_n, Y_n) \\ k_2 &= F(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_1) \\ k_3 &= F(t_n + \frac{h}{2}, Y_n + \frac{h}{2}k_2) \\ k_4 &= F(t_n + h, Y_n + hk_3) \end{aligned}$$

The next value Y_{n+1} is then computed by the sum of the previous value Y_n and a weighted average of the increments:

$$Y_{n+1} = Y_n + \frac{h}{2}(k_1 + 2k_2 + 2k_3 + k_4)$$

Index

- Anchor, 24, 77, 79, 81–87, 89, 91–93, 99–101, 115, 116
- As-Rigid-As-Possible (ARAP), 45, 47, 48, 70, 77, 78, 80, 81, 85, 86, 89–93, 115
- Differential coordinates, 80, 81, 90, 91
- Dynamic mesh, 18–20, 33, 58, 77, 107, 117
- Energy, 41, 56, 78, 79, 85–87, 90–93, 101, 115
- Euclidean distance transform (EDT), 59, 60, 65, 83, 89–91, 100, 101, 116, 137, 138
- Gaussian, 39, 62, 70, 71, 141, 144
- Gradient, 33, 55, 57, 59, 89, 144
- Laplacian, 80–82, 86, 87, 93, 147
- Local rigidity, 23, 41, 49, 79, 80, 86, 91–93, 115
- Minimization, 41, 56, 79, 87, 90, 92, 93, 115, 145
- Model-based, 12, 34, 41–46, 48, 49, 58, 83, 101, 106, 107, 115, 118
- Model-free, 12, 16, 18–20, 22, 33, 34, 37, 39, 46–49, 55, 56, 58, 78, 100, 104, 107, 110, 116
- Motion flow, 23, 24, 49, 55–58, 62–71, 77, 81, 84–86, 92, 93, 99–101, 115, 116
- Optimization, 23, 24, 36, 45–47, 49, 77, 81, 89–93, 100, 101, 115, 116
- Post-production, 20, 21, 23, 55, 58, 99, 110
- RECOVER 3D, 14–18, 20–22, 29, 32, 33, 38, 39, 48, 64, 99, 104, 110, 115–117
- Registration, 14, 40, 41, 44, 47–49, 56, 79–81, 94, 109, 115
- Regularization, 24, 48, 62, 65, 67–70, 78, 89, 90, 93, 101
- Sampling, 19, 20, 43, 44, 47, 83, 84, 94, 100, 109, 116, 117, 139, 140
- Solver, 87, 88, 101
- Temporal consistency, 9, 20, 33, 56, 107, 117

REPRESENTATION DYNAMIQUE DE MODELES D'ACTEURS ISSUS DE RECONSTRUCTIONS MULTI-VUES

Les technologies de reconstruction multi-vues permettent de réaliser un clone virtuel d'un acteur à partir d'une simple acquisition vidéo réalisée par un ensemble de caméras à partir de multiples points de vue.

Cette approche offre de nouvelles opportunités dans le domaine de la composition de scènes hybrides mélangeant les images réelles et virtuelles.

Cette thèse a été réalisée dans le cadre du projet RECOVER 3D dont l'objectif est de développer une chaîne de production TV complète, de l'acquisition jusqu'à la diffusion, autour de la reconstruction multi-vues. Cependant la technologie utilisée dans ce contexte est mal adaptée à la reconstruction de scènes dynamiques.

En effet, la performance d'un acteur est reproduite sous la forme d'une séquence d'objets 3D statiques qui correspondent aux poses successives du personnage au cours de la capture vidéo.

L'objectif de cette thèse est de développer une méthode pour transformer ces séquences de poses en un modèle animé unique.

Les travaux de recherches menés dans ce cadre sont répartis en deux étapes principales. La première a pour but de calculer un champ de déplacements qui décrit les mouvements de l'acteur entre deux poses consécutives.

La seconde étape consiste à animer un maillage suivant les trajectoires décrites par le champ de mouvements, de manière à le déplacer vers la pose suivante.

En répétant ce processus tout au long la séquence, nous parvenons ainsi à reproduire un maillage animé qui adopte les poses successives de l'acteur.

Les résultats obtenus montrent que notre méthode peut générer un modèle temporellement cohérent à partir d'une séquence d'enveloppes visuelles.

Reconstruction multi-vues, animation, déformation de maillage, champ de déplacements, as-rigid-as-possible, scène dynamique, volume, appariement de voxels, cohérence temporelle.

DYNAMIC REPRESENTATION OF ACTORS' MODELS FROM MULTI-VIEW RECONSTRUCTIONS

4D multi-view reconstruction technologies are more and more used in media production due to their abilities to produce a virtual clone of an actor from a simple video acquisition performed by a set of multi-viewpoint cameras.

This approach is a major advance for the composition of animations which mix virtual and real images, and also offers new possibilities for the rendering of such complex hybrid scenes.

The work described in this thesis takes parts in the RECOVER 3D project which aims at developing an innovative industrial framework for TV production, based on multi-view reconstruction, from studio acquisition to broadcasting.

The major drawback of the methods used in this context is that they are not adapted to the reconstruction of dynamic scenes. The output are time series which describe the successive poses of the actor, figured as a sequence of static objects.

The goal of this thesis is to transform these initial results into a dynamic 3D object where the actor is figured as an animated character.

The research detailed in this manuscript presents two main contributions.

The first one is centered on the computation of a motion flow which represents the displacements occurring in the reconstructed scene between two consecutive poses.

The second one presents a mesh animation process that leads to the animation of a 3D model from one pose to another, following the motion flow.

This two-step operation is repeated throughout the entire pose sequence to finally obtain a single animated mesh that matches the evolving shape of the reconstructed actor.

Results show that our method is able to produce a temporally consistent mesh animation from various sequences of visual hulls.

Multi-view reconstruction, animation, mesh deformation, motion flow, as-rigid-as-possible, dynamic scene, volume, voxel matching, temporal consistency.

Discipline : INFORMATIQUE

