Multi-view Reconstruction and Texturing

Ehsan Aganj

Ecole Nationale des Ponts et Chaussées

Partially supported by Flamenco ANR Project

Doctorate of Science

COMPUTER SCIENCE

Multi-view Reconstruction and Texturing

Ehsan AGANJ

Thesis advisor: Renaud KERIVEN

CERTIS , IMAGINE

Thesis defended on December 11, 2009

Jury :

Dror AIGER	ESIEE/LIGM	Invited member
Edmond BOYER	INRIA, PERCEPTION	Reviewer
Renaud KERIVEN	IMAGINE/LIGM	Advisor
Laurent NAJMAN	ESIEE/LIGM	Examiner
Jean-Philippe PONS	IMAGINE/CSTB	Co-advisor, Invited member
Mariette YVINEC	INRIA, Sophia Antipolis	Reviewer

To my parents...
and my brother...

Abstract

Abstract In this thesis, we study the problem of static and dynamic multiview reconstruction and texturing, particularly focusing on real applications. First, we propose three reconstruction methods sharing the objective of estimating a representation of a static/dynamic scene from a set of multiple images/videos. Then, we consider the problem of multi-view texturing, focusing on the visual correctness of the rendering. The contributions of this thesis are as follows,

- A shape from silhouette approach is proposed producing a compact and high-quality 4D representation of the visual hull, offering easy and extensive control over the size and quality of the output mesh as well as over its associated reprejection error.
- A dynamic multi-view reconstruction method is proposed computing a 4D representation of a dynamic scene, based on a global optimization of a true spatio-temporal energy, taking visibility into account.
- A photo-consistent surface reconstruction method is proposed incorporating the input images for better accuracy and robustness.
- A multi-view texturing method is proposed computing a visually correct texturing of an imprecise mesh.

Résumé Dans cette thèse, nous étudions les problèmes de reconstruction statique et dynamique à partir de vues multiples et texturation, en s'appuyant sur des applications réelles et pratiques. Nous proposons trois méthodes de reconstruction destinées à l'estimation d'une représentation d'une scène statique/dynamique à partir d'un ensemble d'images/vidéos. Nous considérons ensuite le problème de texturation multi-vues en se concentrant sur la qualité visuelle de rendu.

Keywords Multi-view reconstruction, dynamic reconstruction, stereovision, surface reconstruction, point cloud, Delaunay triangulation, Voronoi diagram, medial axis transform, cell complex, minimum s-t cut, simulated annealing, visibility, thin-plate spline, texturing

Remerciements

En premier lieu, je tiens à adresser mon entière reconnaissance à mon directeur de thèse, Renaud Keriven, pour m'avoir accueilli au sein de son laboratoire et avoir dirigé les travaux présentés dans ce manuscrit. J'adresse également toute ma gratitude à Jean-Philippe Pons et à Florent Ségonne qui ont fortement influencé la direction prise par cette thèse.

Tous mes remerciements aux membres de mon jury de soutenance de thèse pour avoir accepté d'y participer, plus particulièrement à Edmond Boyer et Mariette Yvinec pour avoir accepté d'être rapporteur de ce manuscrit malgré la charge de travail que cela représente.

Un grand merci à tous les membres de l'équipe IMAGINE, de l'ancien laboratoire CERTIS, pour tout ce qu'ils m'ont apporté, aussi bien scientifiquement et humainement qu'à tout autre niveau. Je remercie particulièrement Pascal Monasse pour les échanges et collaborations scientifique qui apparaissent dans la deuxième partie de cet ouvrage.

Je remercie l'Ecole des Ponts et l'Université Paris-Est pour avoir assuré l'organisation administrative et logistique de ma thèse ainsi que l'Ecole Polytechnique et le Rectorat de Paris, m'ayant fourni la bourse nécessaire au déroulement de la thèse.

Je dois finalement remercier mes parents et mon frère pour leur soutien sans faille: merci infiniment à vous.

Contents

Notations

$\mathbb{E}^d$	Euclidean space of dimension d
$\mathbb{R}^d$	real space of dimension d
$conv()$	convex hull
$c(p,q)$	capacity of an edge connecting p and q in a graph
d	dimension of the space
$d(X,Y)$	Euclidean distance between X and Y
$d(X,\Omega)$	Euclidean distance between X and Ω (a subset of $\mathbb{R}^d$)
$d_{\text{pow}}(\Sigma_1,\Sigma_2)$	power distance between spheres Σ_1 and Σ_2
$d_{\text{prob}}(X,Y)$	probabilistic distance between X and Y
i	index variable
$\vec{l}$	labeling vector
k	index variable
n	number of objects, cardinality of a set of objects
p,q	nodes of a graph, probability
r	radius of a sphere or a ball
s	state
t	time
v	scaling factor, node
$(x_1,x_2,\cdots,x_d)$	real coordinates of a point in $\mathbb{R}^d$
$\mathcal{B}$	set of balls
$\mathcal{C}$	complex, cut, set of cells
$\mathcal{D}(\mathcal{M})$	Delaunay polytope, dual to $\mathcal{V}(\mathcal{M})$
$\mathcal{D}el(\mathcal{M})$	Delaunay complex of $\mathcal{M}$
$\mathcal{D}el\|_\Omega(\mathcal{M})$	Delaunay triangulation of $\mathcal{M}$ restricted to Ω
$\mathcal{E}$	set of edges in a graph
$\mathcal{G}$	graph
$\mathcal{H}$	set of hyperplanes
$\mathcal{I}$	set of images
$\mathcal{M}$	set of points
$\mathcal{P}$	polytope, paraboloid, set of probabilities
$\mathcal{P}ow(\mathcal{S})$	power diagram of a set $\mathcal{S}$

$\mathcal{S}$	surface, curve, subset of nodes of a graph containing the source, set of spheres
$\mathcal{T}$	subset of nodes of a graph containing the sink
$\mathcal{V}$	set of nodes in a graph
$\mathcal{V}or(\mathcal{M})$	Voronoi diagram of a set $\mathcal{M}$
B	ball
$B(C,r)$	ball of center C and radius r
C	cell, center of a sphere or a ball
E	energy
F	facet
H	hyperplane
$\tilde{H}$	half-space
I	image
M	vertex, point, vector
$O()$	asymptotic upper bound
P,Q	vertex, point, vector
R	constant
T	temperature
U	vertex, point, vector
V	visual hull
$V(M)$	Voronoi cell of a point M
X,Y	vertex, point, vector
$\phi()$	signed distance function in $\mathbb{R}^2$
γ	photo-consistency measure
ρ	probability density function
$\Phi()$	signed distance function in $\mathbb{R}^3$ and $\mathbb{R}^4$
$\Pi()$	camera projection
$\Theta()$	asymptotic equivalent
Σ	sphere
$\Sigma(C,r)$	sphere of center C and radius r
Ω	silhouette, subset of $\mathbb{R}^d$

Introduction

Reconstructing models of a scene or an object from multiple images, taken from different viewpoints, has long been of interest in the area of computer vision. More recently, the automatic spatio-temporal modelization of a dynamic scene is becoming increasingly popular. A large number of applications require good estimation of shape and/or motion. In medicine, the shape and the motion of organs and tissues are modeled to study anatomical structures, diagnose disease, plan treatment or guide and monitor surgery. In sport science, the body parts of athlets are tracked to improve their performance. In game and movie industry, human performance is captured and mapped on photo-realistic reconstructed models of virtual actors. Today's increasingly demanding applications for these problems present several challenges for computer vision techniques.

Controlled Quality and Precision

Despite highly improving rate of the capabilities of digital electronic devices, more applications require accesible control over the quality and the precision of the representation, aiming generally to handle computational complexity as a function of the available resources. Generally speaking, two major directions have been followed: The first has been focused on the precision and/or quality of the reconstruction, while the second has given up the accuracy focusing on real-time processing. Very few works provide a complete control over the quality and the accuracy in a way to give the ability of tuning the performence and the precision.

Compactness

Increasingly growing resolution of aquisition devices yields massive digitized scene representations. On the other hand, more and more applications require efficient ways of stocking and transfering data, particularily because of the growing speed of virtual/online environments. Yet, not many approaches have been proposed considering compactness as a crutial objective, most algorithms produce large meshes which should be compressed or simplified in a post-processing step.

Coherent Representation

A frame-by-frame reconstruction of a dynamic scene does not exploit temporal coherency, leaving discontinuities and visual artifacts. Most applications, however, demand compact and realistic representations. Such results are only attainable when the spatial reconstruction is guided by temporal cues, exploiting the sequence in a global manner.

Robustness

No real scanning devise provides exact data. This is particularily a challenging problem for the surface reconstruction and multiview reconstruction algorithms based on point clouds extracted from image datasets, featuring higher levels of noise and higher proportions of outliers than those of active scanning devices. Unfortunatily, this discards most, if not all, standard surface reconstruction algorithms, and an important part of multiview reconstruction methods. In practice, a reconstruction approach cannot be employed in real applications unless it robustly handles uncertainty.

Visual Correctness

Some applications desire visually correct rendering of a scene, despite the noticable imprecision of the representation (e.g. entertainment industry). Generally as a last stage of the pipeline, no farther correction of the mesh is possible (e.g. the origin of error is unknown, mesh is a visual hull). Most proposed approaches, however, do not consider the visual correctness an important factor, very few works have been done in this direction.

Contributions of this thesis

In this thesis, we study the problem of static and dynamic multiview reconstruction, followed by a texturing step, generally employed in most applications to obtain a complete realistic computer representation of the scene. We are particularly focusing on the aforementioned motivations, specially important for real applications. We propose four novel approaches,

- The shape from silhouette approach presented in Chapter 3 produces a compact and high-quality 4D representation of the visual hull, offering easy and extensive control over the size and quality of the output mesh as well as over its associated reprojection error.

- The dynamic Multi-view reconstruction method presented in Chapter 4 computes a 4D representation of a dynamic scene, based on a global optimization of a true spatio-temporal energy, taking visibility into account.
- The photo-consistent surface reconstruction method presented in Chapter 5 incorporates the input images for better accuracy and robustness.
- The multi-view texturing method presented in Chapter 6 computes a visually correct texturing of an imprecise mesh.

The first part of this thesis consists in our contributions for the problem of static and dynamic multi-view reconstruction. We consider all over this thesis that images/videos are captures from calibrated multi-view cameras defined by projections $\Pi_i : \mathbb{R}^3 \to \mathbb{R}^2$ from world reference frame to the image planes. In the dynamic case, we also suppose that the cameras are fixed and the projections are not modified through time. Two situations have been distinguished according to the nature of input images:

1. the input is a set of silhouettes (Chapter 3)
2. the input is a set of color images (Chapters 4 and 5)

In chapter 5 we suppose that the point clouds have already been extracted from images by triangulation. However, we are still in a multi-view framework and we propose a photo-consistent surface reconstruction algorithm.

In the second part of this thesis, we study the problem of multi-view texturing, often considered as the last stage of the pipeline. Our objective is to compute a visually correct rendering: we propose an additional step to make sure that images correspond to each other according to the mesh.

The remainder of this thesis is organized as follows.

Chapter 1

In this chapter we review the problem of shape from silhouette and the problem of dynamic multi-view reconstruction. We survey the most related existing techniques to our work, and describe the different directions followed in this domain.

Chapter 2

In this chapter we give some background on the basic computational geometry concepts needed in this thesis: Convex hulls, Polytopes, Simplices, and Complexes 2.1, Voronoi Diagram 2.2, and Delaunay triangulation 2.3.

Chapter 3

In this chapter we present a novel method for computing a four-dimensional representation of the spatio-temporal visual hull of a dynamic scene. We give some background on the concept of Restricted Delaunay triangulation 3.2, we describe our method for static and dynamic visual hull reconstruction 3.3, and we report on some numerical experiments which demonstrate the effectiveness and flexibility of our approach for generating compact, high-quality, time-coherent visual hull representation from real silhouette image data.

Chapter 4

In this chapter we present a novel method for computing a globally optimal spatio-temporal representation of a dynamic scene from videos. We give some background on the energy minimization via graph cuts 4.2, we describe our method for multi-view reconstruction from videos adapted to dynamic cluttered scenes 4.3, and we report some numerical experiments which demonstrate the potential of our approach to reconstruct real dynamic scene under uncontrolled imaging conditions 4.4.

Chapter 5

In this chapter we present a novel method for the problem of surface reconstruction robust to noise and outliers. We give some background on the concept of Power Diagram, Regular triangulation, medial axis transform, poles and polar balls, and the optimization method of simulated annealing 5.2, we describe our method incorporating the photometric cues with the surface reconstruction process for a better accuracy and robustness 5.3, and we report numerical experiments which compare it favorably with the state of the art in terms of accuracy and robustness 5.4.

Chapter 6

In this chapter we present a novel method for texturing of imprecise mesh. We give some background on the algorithm thin-plate splines 6.2.4, we describe our method for visually correct texturing based on the deformation of input images 6.2, and we report on some numerical experiments validating our method on simulated and real imprecise meshes 6.3.

Publications of the author

- E. Aganj, J.-P. Pons, F. Ségonne and R. Keriven. *Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing.* In IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, Oct 2007.
- E. Aganj, J.-P. Pons and R. Keriven. *Globally optimal spatio-temporal reconstruction from cluttered videos.* In Asian Conference on Computer Vision, 2009.
- E. Aganj, R. Keriven and J.-P. Pons. *Photo-consistent surface reconstruction from noisy point clouds.* In IEEE International Conference on Image Processing, 2009.
- E. Aganj, P. Monasse and R. Keriven. *Multi-View Texturing of Imprecise Mesh.* In Asian Conference on Computer Vision, 2009.

Part I

Multi-view Reconstruction

Chapter 1

Introduction

Abstract

In this introductive chapter, two problems have been reviewed: The problem of shape from silhouettes, and the problem of dynamic multiview reconstruction. In the first part, mainly related to Chapter 3, the problem of shape from silhouette is studied in both static and dynamic cases, surveying the existing techniques which aim to approximate the shape of a scene, or a spatio-temporal shape of a dynamic scene from only silhouette cue. Despite the approximative nature of these representations, the shape from silhouette methods are quite popular in computer vision and graphics due to their computational efficiency and straightforward implementation. In the second part, mainly related to Chapter 4, the problem of dynamic multiview reconstruction has been reviewed in a more general case, surveying the methods which employ photometric and (possibly) silhouette information to reconstruct a spatio-temporal approximation of the scene.

Contents

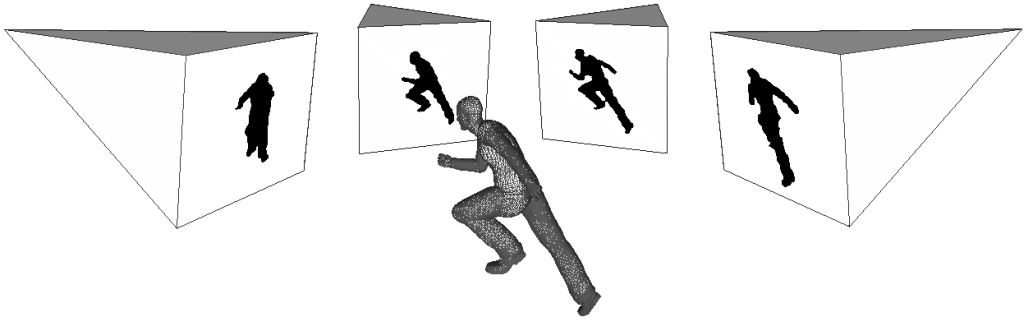


Figure 1.1: Silhouettes of an object with respect to four viewpoints. Silhouettes are shown as binary images.

1.1 Shape from Silhouette

The problem of estimating the shape of an object or a scene from multiple views has received a considerable interest in computer vision. *Shape from silhouette* is one popular class of methods to solve this problem in an approximate but efficient and robust manner. Most of these techniques consist in computing the *visual hull*, which is the maximal volume consistent with a given set of silhouettes. One noticeable exception is the recent work of Franco *et al.* [Franco et al., 2006] which constructs smooth silhouette-consistent shapes strictly included in the visual hull.

Shape from silhouette approaches are quite popular in computer vision and graphics due to their computational efficiency and straightforward implementation, especially compared to the photometric reconstruction methods such as stereo. These methods are widely used in modeling and localization applications such as human modeling, interactive virtual environments, virtualized reality, and motion tracking.

The first known shape-from-silhouette approach was presented by [Baumgart, 1974]. The concept of visual hull was then introduced by Laurentini [Laurentini, 1994], and it has been extensively studied since then. A formal definition of the visual hull of a scene relative to a set of viewpoints is given as follows.

1.1.1 Definition

Definition 1. *A silhouette is the 2D projection of a 3D object or a 3D scene to the image plane. A silhouette image is a binary image indicating at each image point if the back-projected ray of that point intersects or not the object.*

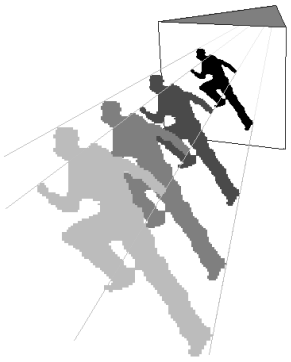


Figure 1.2: A viewing cone. The infinite cone starts from the viewpoint and passes through the contour of the silhouette.

Figure 1.1 shows an example of silhouettes of a 3D object with respect to four viewpoints. A point P of the image plane is inside the silhouette if the ray starting at the viewpoint and passing through P intersects the object. In this example silhouettes are shown as binary images where black foreground and white background represent pixels inside and outside the silhouette respectively.

Several reasons give an important role to the silhouette in computer vision. The silhouette of an object is an effective cue to understand its shape [Aloimonos, 1988]. It is widely used in the problems of 3D modeling, object identification [Besl and Jain, 1985, Chien and Aggarwal, 1989, Wang et al., 1984], multi-object localization and motion capture. Silhouettes can be efficiently computed by thresholding algorithms.

Definition 2. *A viewing cone is the back projection of the silhouette to space. It is the maximal region of space consistent to it.*

As seen in figure 1.2 the infinite cone of a silhouette is a generalized cone that originates from the viewpoint and passes through the silhouette's contour on the image plane. Any point of space in the obtained region can possibly belong to the object. The visual hull of a collection of silhouettes is then defined as the intersection of their cones. We note that the visual hull of an object is generally defined as the maximal volume consistent with the silhouettes on all possible views. However, as in practice only a finite number of views are provided, the visual hull is usually refered to the region obtained by their intersection. Please refer to [Koenderink, 1984] for some relationships between the occluding contour and the surface of a shape. A formal definition of the visual hull follows,

Definition 3. *Let* $\mathcal{I} = \{I_1, \cdots, I_n\}$ *be the set of input images, and let* $\Pi_i : \mathbb{R}^3 \to \mathbb{R}^2$ *denote the camera projections from the world reference frame to the image planes.*

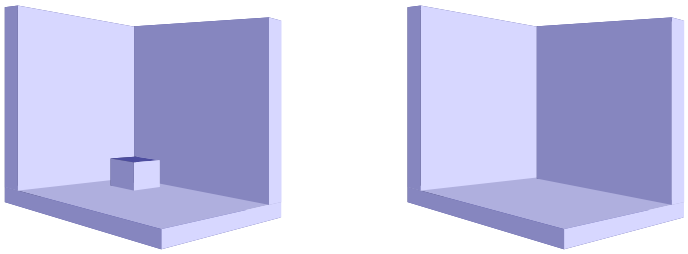


Figure 1.3: Two objects which have the same silhouettes.

In addition, $\Omega_i \subset \mathbb{R}^2$ *are the silhouettes of the object in the different images. The visual hull* V *is then defined by:*

$$V = \left\{ P \in \mathbb{R}^3 \mid \forall i \in \{1, \ldots, n\},\ \Pi_i(P) \in \Omega_i \right\} . \tag{1.1}$$

Figure 1.4 shows the visual hull of the object with respect to the four silhouettes. The visual hull approximates the shape of the object capturing all geometric information given by its silhouettes. The approximation depends on the complexity of the shape, as well as the number of views, their positions and their orientations. The object is guaranteed to lie in its visual hull. However, the visual hull does not uniquely determine its shape. An infinite set of shapes would give the same silhouettes, for example in the case of concavities in the object. Figure 1.3 shows an example of two objects which have the same visual hull. Please refer to [Boyer and Franco, 2003] for formal and detailed definitions of contours, viewing cones and visual hulls.

1.1.2 State of the art

Visual hull algorithms fall into three main categories: *volume-based* approaches, *surface-based* approaches, and *image-based* approaches.

1.1.2.1 Volume-based methods

Volume-based methods use a subdivision of space into elementary regions, typically voxels. The voxel is the quantum unit of volume representing a value on a grid in three-dimensional space. This is analogous to the two-dimensional pixel. In most volume-based methods voxels are labeled by numerical values representing different aspects of the scene. An early approach which used this representation was [Martin and Aggarwal, 1983]. In their work parallelepipedic cells aligned with the coordinate axis are used as the volume elements.

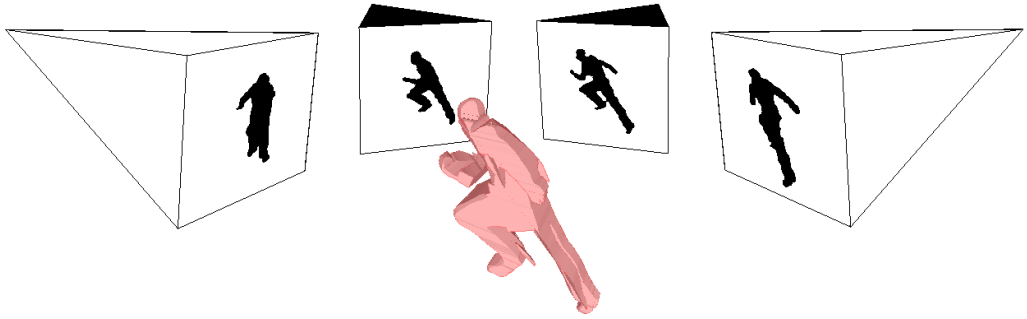


Figure 1.4: The visual hull of the object in figure 1.1 with respect to four silhouettes (Visual hull computed by the method of [Franco and Boyer, 2003]).

Voxel occupancy is the simplest volume-based approach to this problem. In voxel occupancy each voxel is labeled as filled or empty, representing the scene as the set of filled voxels. Many approaches are based on this representation. The classical shape-from-silhouette algorithm which uses this representation is called *silhouette intersection*. In this method the cone of each silhouette is computed as the set of all voxels which may be occupied. The 3D shape is then computed as the intersection of the cones giving the collection of all voxels consistent with the silhouettes. A classic algorithm known as *voxel carving* computes this intersection by projecting each voxel to all viewpoints and discarding all voxels which fall outside any of the silhouettes. Several methods have been proposed to compute and represent efficiently the shape. In [Potmesil, 1987, Szeliski, 1993] the octree representation (a tree of recursively subdivided cubes) is considered handling arbitrary viewspoints.

It is important to note that silhouettes are considered as hard constraints (i.e. the projections of voxels must lie into the silhouettes). In practice this is not a good assumption since small errors in the silhouettes can have dramatic effect on the result (i.e. a pixel error makes a hole in the object). In this case, the voxel carving approach yields very noisy reconstructions by making hard decisions which cannot be recovered, and by discarding the spatial coherence between voxels. Unfortunately silhouette extraction is generally a difficult task to achieve in presence of real world perturbations. This is why silhouette error robustness has become an important issue in the problem of shape-from-silhouette.

Several works have been done to overcome this problem. In most of them the simultaneous information from all images are integrated to make more relevant decisions. In [Snow et al., 2000] a generalization of silhouette intersection is proposed. In their method a binary labeling of the voxels is computed by finding the global

minimum of an energy dealing with a data term and a smoothness term. Silhouettes' hard constraint are replaced with a soft constraint: First, silhouettes are not computed exclusively. This avoids an important source of error coming from the silhouette computation. Second, a global spatial smoothness is incorporated. The data ambiguity is resolved taking advantage of the value consistency on neighboring voxels. The global minimum of their energy is then computed using graph cuts.

In a more recent work [Franco and Boyer, 2005] a probabilistic representation of the problem is proposed. Unlike the binary formulation of voxel occupancy the scene is represented by a grid of voxel occupancy probabilities. Similarly to the previous method they compute silhouette fusion in space in order to integrate the contribution of all images. This is done by considering each pixel of input images as a statistical occupancy sensor. The result of their method is then a global probability inferred using all the pixel observations in each voxel. This encodes naturally the shape information.

Graphics hardwares are also used in volume-base methods to compute efficiently the shape approximation. In [Lok, 2001] projected textures are used to decide if a volume sample lies within the visual hull. Using framebuffer their method is implemented on graphics hardware making the real-time reconstruction possible. An application of their method is in the immersive virtual environments (IVE).

Volume-based methods suffer generally from quantization and aliasing artefacts, as well as from a high memory cost. Moreover, additional morphological processings are usually needed in order to make correct result [Serra, 1983]. When a final mesh representation is required, these methods must be complemented with an isocontour extraction algorithm, such as the *marching cubes* algorithm, introduced by Lorensen and Cline [Lorensen and Cline, 1987]. Unfortunately, this technique produces unnecessarily large meshes (at least one triangle per boundary voxel) of very low quality (lots of skinny triangles). Frequently, the resulting meshes have to be regularized, optimized and decimated in order to obtain suitable representations in terms of compactness and quality, while simultaneously controlling the approximation accuracy and preserving some topological properties, such as the absence of self-intersections, which turns out to be a difficult task.

1.1.2.2 Surface-based methods

Surface-based methods consist in directly building a mesh representation of the visual hull by computing the intersection of viewing cone surfaces associated with the silhouettes. In an early approach [Baumgart, 1975] polyhedral surfaces were used to represent the shape approximation. Several works have been done since then dealing with surface information [Koenderink, 1984, Cipolla and Blake, 1992,

Boyer et al., 1995, Sullivan and Ponce, 1998].

In [Matusik et al., 2001] a polyhedral representation of the visual hull is computed. In their method all silhouette cones are intersected to make a set of polygons that defines the surface of the visual hull. 3D intersections are reduced to simpler intersections in 2D thanks to the fact that silhouette cones have constant scaled cross-section. The intersection of a face of a cone with another cone is computed by projecting the face on the image plane of the other cone, computing its 2D intersection with the silhouette, and back-projecting the result on the plane of the face. Edge-Bin data structures (described in [Matusik et al., 2000]) are used to accelerate the intersection process. Their method computes the visual hull quickly and can be used in real-time applications.

In [Lazebnik et al., 2001] the visual hull is computed as a generalized polyhedron where faces, edges and vertices are formed by the intersections of the silhouette cones. In their algorithm these features are correctly connected to form a precise topological and geometric mesh. Their algorithm uses two-dimensional computations and works with weakly calibrated cameras taking advantage of epipolar geometry.

In [Boyer and Franco, 2003] a hybrid approach is proposed which takes advantage of surface-based representation to overcome the precision limits of the volume-based approach. In their method the regular voxel grid of the volume-based methods is replaced by an irregular space discretization where vertices lie on the surface of the visual hull. A Delaunay triangulation is computed on the sample points making a collection of tetrahedral cells. Delaunay cells are then carved according silhouette information to approximate the visual hull. The proposed hybrid approach remains efficient and robust taking advantage of the volumetric representation while giving precise results with lower time and space complexities.

In a following work [Franco and Boyer, 2003] the method of [Boyer and Franco, 2003] is modified to compute effeciently an exact polyhedral visual hull associated to polygonal image contours. In their work the Delaunay computation step used in [Boyer and Franco, 2003] is replaced by an algorithm that recovers mesh connectivity. This yields to a polyhedron that projects correctly on the silhouettes. Their method is low in time complexity and suitable for real-time applications.

An advantage of surface-based approach compared to the volume-based approach is the eventual precision of the visual hull computation. However, it is generally admitted that surface-base algorithms suffer from numerical instability in the case of noisy or mutually inconsistent silhouette data. Moreover, the output mesh has a very low quality, and typically develops degeneracies when the number of input views increases. More generally, in these approaches, there is no easy way to control

the size or the quality of the output mesh, because it is related in a very complex way to the chosen polygonization of image contours.

1.1.2.3 Image-based methods

For sake of completeness, let us also mention the image-based approach of Matusik *et al.* [Matusik et al., 2000] that generates on-the-fly a view-dependent visual hull representation for the purpose of rendering. Contrarily to the volume-based and surface-based methods, they do not reconstruct a geometric or volumetric representation. Depending on the rendering viewpoint, the visual hull is computed in the image space of the reference images.

The advantage of this approach is that sampling is done only on the pixels of the desired image. The exact image-based representation is rendered on a given view by computing the intersections of the discrete viewing rays with the visual hull. These intersections can be computed efficiently on the 2D image planes taking advantage of the epipolar geometry. The proposed approach has low complexity in time and is shown to be suitable for real-time applications.

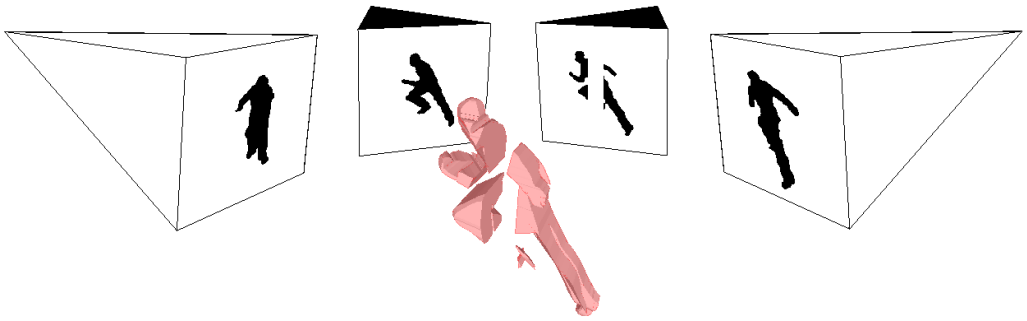


Figure 1.5: The visual hull of the object in figure 1.1 with respect to four silhouettes (Visual hull computed by the method of [Franco and Boyer, 2003]). One of the silhouettes is partially occluded.

1.1.2.4 Occluded silhouettes

Most background extraction methods fail to extract correctly the silhouette in presence of occlusions. Using these incomplete silhouettes the visual hull methods produce incomplete shape which do not contain the actual object. Figure 1.5 shows the case where one of the silhouettes is occluded. In order to avoid this situation,

partially occluded images are usually discarded even if they contain useful shape information in the non-occluded parts.

In a recent work [Guan et al., 2006] an extended visual hull method is proposed to deal with partial occlusions. In order to make a complete visual hull, they propose to compute extended silhouettes which are guaranteed to contain the real silhouettes of the object. This is done using extracted silhouettes and binary occlusion masks which identify occluded pixels. These masks are extracted by an algorithm based on a novel concept of effective boundary using spatiotemporal information. The visual hull obtained by their algorithm is guaranteed to contain the real shape.

1.1.2.5 Dynamic shape-from-silhouette

While many authors have focused on computing the visual hull in the case of static images, leading to several established techniques mentioned above, very little work has dealt with the case of dynamic scenes captured by multiple video sequences, from an actual spatio-temporal perspective, i.e. by going beyond independent frame-by-frame computations.

The most notable related work is that of Cheung *et al.* [Cheung et al., 2005a] on computing visual hull across time. However, their purpose is not to reconstruct a dynamic scene. Their goal is to virtually increase the number of input silhouettes by registering many views of a rigidly-moving object in a common reference frame, in order to approximate the shape of the object more accurately. The virtual views are computed by estimating the motion of the object between successive frames. Unfortunately, the motion cannot be uniquely determined from only the silhouette information. In order to resolve the alignment ambiguity, they propose to combine color information with silhouette cue to extract a point cloud on the surface for each frame. The motion can then be computed by aligning the point sets. See also [Vijayakumar et al., 1996] for an early work in this direction.

More related to the dynamic reconstruction is the extension of [Cheung et al., 2005a] to articulated motion [Cheung et al., 2005b]. Under the articulated assumption, their approach exploits temporal coherence to improve shape reconstruction while estimating the skeleton of the moving object. However, their approach is experimentally validated in the case of a single joint only. Hence it is not relevant to the case of complex human motion or non-rigid scenes.

In a recent work [Vlasic et al., 2008] more complex human motion is handled using silhouette geometric information. In their method visual hull is used to capture the motion of a human performer in order to make a detailed mesh animation. First, the skeletal pose is tracked, then an articulated template is deformed to fit the recovered pose at each frame. The obtained meshes are therefore in full

correspondence and have correct topology. This is particularly suitable for the mesh editing processes such as texturing. However, their method is limited to articulated objects and produces incorrect geometry in presence of silhouette errors.

More works have been done in the domain of dynamic reconstruction. However, most of them are based on the photometric information and do not reconstruct the visual hull approximation. Hence, they do not belong to the shape-from-silhouette category and are described in a separated section 1.2.

1.2 Dynamic Multi-view reconstruction

Spatio-temporal modeling of a dynamic scene is a challenging problem which has recieved a great attention in computer vision. In recent years, several methods for automatic generation of complete models from multiple videos have been proposed. In particular, the most recent ones have proven effective for full-body marker-less motion capture of non-rigid scenes. However, many approaches make assumptions on the shape, among which the rigidity or piecewise-rigidity of the scene. In the following, different varieties of the problem is discussed. Existing approaches are reviewed describing different directions made to deal with dynamic scenes.

1.2.1 State of the art

Dynamic multi-view reconstruction is generally defined as the problem of computing a spatio-temporal model of a dynamic scene from multiple synchronized videos captured from different view-points. Specific scenarios can although be considered making a large variety of situations depending on the *a priori* knowledge of the scene and the motion. One important example is the case of rigid objects [Ullman, 1979, Waxman and Duncan, 1986, Young and Chellappa, 1990, Zhang and Faugeras, 1992, Costeira and Kanade, 1998] where the structure and the motion can even be computed only from a single video sequence. This is due to the fact that the projection of a moving rigid object on a camera has a highly constrained two-dimensional motion. As an example, motion capture methods based on kinematic skeleton rely on piecewise rigid motion. These methods, however are not of interest here, since generally they do not recover spatio-temporal details, and fail to track non-rigid motions. Please refer to [Moeslund and Granum, 2001] for a survey on this domain.

The dynamic reconstruction is of great interest in a large number of applications such as object recognition, medecine, kinesiology and cinema. Specific applications, such as cardiac motion, can make parametric models to constraint the shape and the motion. Nevertheless, most recent works are focused on the general motion of a complex scene dealing with important challenges such as topology changes or self

occlusions.

A challenging task in the problem of dynamic reconstruction is the matching problem of same physical points in the spatio-temporal framework. Particularly, a large number of tasks and applications, such as compression or spatio-temporal post-processing, require dynamic models with constant spatial connectivity over time, relying directly on the problem of correspondence finding. Unfortunately, this is a very difficult task since a three-dimensional patch has generally very different shapes and colors when seen from different views or through time. This requires image similarity measures capable of finding correspondences in arbitrary situations. To simplify the matching process, however most approaches make restricting assumptions on the appearance of the surface patches. A popular example is the assumption that the corresponding pixels have the same color in all views and in all time instants. This is clearly not valid in the case of non Lambertian surfaces. Moreover, temporal correspondences cannot be found correctly under this assumption unless the images are taken under controlled environments such as constantly illuminated scenes.

Depending on the application, more robust matching criterions have been also considered. Some important examples are the normalized cross correlation [Faugeras and Keriven, 1999, Goldlücke and Magnor, 2004, Esteban and Schmitt, 2004], radiance tensor [Jin et al., 2005] or statistical measures such as the correlation ratio [Roche et al., 1998] and mutual information [Viola et al., 1995].

1.2.1.1 Scene Flow Estimation

First family of approaches relies on the estimation of *scene flow*, the three-dimensional motion field of a dynamic scene introduced in [Vedula et al., 1999]. This is analogous to the two-dimensional *optical flow*, defined as the motion field of points on an image, naturally related to it by camera projection: the projection of the scene flow on a camera results the optical flow of the two-dimensional projection of the dynamic scene. Indeed, this is not invertible, clearly the projected two-dimensional motion information encoded in optical flow does not give enough information about the three-dimensional motion of a scene, more knowledge about the structure of the scene and the rates of change of depth maps are required to compute the scene flow from optical flow of a single camera [Vedula et al., 1999].

Definition 4. *The optical flow is the two-dimensional motion field of points on an image. Let* $\vec{U}_i(t) : \mathbb{R}^+ \to \mathbb{R}^2$ *be the temporal trajectory of a moving point on plane, its optical flow is* $\frac{dU_i}{dt}$*. Similarily, the scene flow is the three-dimensional motion field of points in space. Let* $\vec{X}_i(t) : \mathbb{R}^+ \to \mathbb{R}^3$ *be the temporal trajectory of a moving point in space, its scene flow is* $\frac{dX_i}{dt}$*.*

Suppose that $U_i(t)$ is the projection of the three-dimensional point $X_i(t)$ on the image plane of a camera. Their trajectories are then related by the camera projection $\Pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ from the world reference frame to the image plane:

$$\begin{aligned} U_i &= \Pi(X_i) \\ \frac{dU_i}{dt} &= \frac{\partial \Pi}{\partial X} \frac{dX_i}{dt} \end{aligned} \tag{1.2}$$

The scene flow and the optical flow are therefore related by the 2×3 Jacobian matrix $\frac{\partial \Pi}{\partial X}$ which can be derived by differentiating the camera projection symbolically. Unfortunately, this equation provides only two constraints and can not be used to find the three unknowns of the inverse estimation problem of scene flow from optical flow. However, as proposed in [Vedula et al., 1999], multiple optical flows captured by multiple cameras yield to the following over-constrained system which can be solved by a singular value decomposition of the Jacobian matrix. This yields to the least square minimization of the error obtained by reprojecting the scene flow on each optical flow:

$$\begin{pmatrix} \frac{\partial \Pi_1}{\partial X} \\ \vdots \\ \frac{\partial \Pi_n}{\partial X} \end{pmatrix} \frac{dX}{dt} = \begin{pmatrix} \frac{\partial U_1}{\partial t} \\ \vdots \\ \frac{\partial U_n}{\partial t} \end{pmatrix} \tag{1.3}$$

A limitation of this method is the reliance to the accuracy of optical flow estimation, as it is computed separately. In a more recent work [Vedula et al., 2005], this result is used in a pipeline over two earlier works [Vedula et al., 1999, 2002] about scene flow estimation and image interpolation. In their work an image-based approach is proposed to model dynamic events and render novel virtual views of the scene. The 4D spatio-temporal model of their approach allows to interpolate the scene from arbitrary viewpoint at any time. Their pipeline consists in two steps: First, the shape and the scene flow is computed using photometric information making a 4D model of the scene and the motion. Then a spatio-temporal view interpolation algorithm is used to generate novel image from arbitrary spatio-temporal viewpoint.

The shape of the scene at each time instant is first estimated individually in a volumetric framework as a 3D voxel model by the voxel coloring algorithm [Seitz and Dyer, 1997]. The scene flow is then estimated using multiple optical flows from multiple cameras as described above. In the second step, a new image is rendered from arbitrary viewpoint by using the motion to 'flow' the nearest computed shape to an arbitrary time instant, fitting a smooth surface to the obtained voxel model, and finally ray-casting across space and time to generate the image.

Knowledge of the scene flow is very useful in a large number of applications such as motion analysis, motion capture, object recognition and computer animation. Particularly, the velocity distribution can be used as a prediction mechanism to compute the spatio-temporal structure more efficiently and more robustly. Such model can then be used to interpolate novel views across time and space. Many other works have been done to estimate the scene flow, in some of which structure and motion are computed simultaneously exploiting the relationship between structure and motion constraints, this is called *motion stereo*. In [Zhang and Kambhamettu, 2000] the shape and motion estimation are integrated. An affine motion model is fitted to an image grid with a global smoothness constraint on the whole image. Their method, however does not handle correctly the occluded parts and the motion/depth boundaries. This limitation is corrected in an improved version [Zhang and Kambhamettu, 2001] which handles discontinuities and occlusion.

In the volume-based approach of [Vedula et al., 2000] the shape and the scene flow of a non-rigid scene are recovered simultaneously by carving the 6D space of all possible shapes and flows. The shape computation is integrated with the motion estimation using images of two time instants. The advantage of this approach compared to the static shape estimation methods is that the additional information provided by the motion data is integrated in order to recover more robustly the shape and the motion.

The 6D space is formed by *hexel*s, points representing the beginning and ending position of a fixed point on the surface of a moving object in two time instants. A 2D manifold in this space represents a spatio-temporal scene. In order to find this surface, two assumptions are made. First, a point on the scene keeps the same color through time. Second, the motion is bounded between frames. The shape and the motion are computed by carving hexels which do not lie on the spatio-temporal manifold. Their experimental results show that this method improves the shape by integrating the motion, however, it does not guarantee the smoothness of the computed scene flow as it is estimated individually for each voxel. A disadvantage of their method is its computational complexity and memory demands due to the 6D-hexel grid, making it not applicable to high resolutions.

Reflectance, as well as shape and motion of a non-rigid scene is recovered in [Carceroni and Kutulakos, 2002]. They overcome the Lambertian reflectance constraint of previous approaches by assuming a Phong model, handling shadows and specular highlights. In another surface-based approach [Neumann and Aloimonos, 2002] both motion and structure fields of a dynamic scene are computed using silhouette and photometric information. The advantage of their approach compared to the volume-based approach of [Vedula et al., 2000] is that a multi-resolution

subdivision framework is used, locally adapting the surface representation to the complexity of shape and motion in order to obtain high resolution result. Please refer to [Zorin et al., 1997] for a description on the multiresolution representation of meshes used in their work. However, unlike the work of [Zorin et al., 1997] where the user manipulates different levels of resolution, they propose to change the shape and motion by optimizing a multi-view spatio-temporal stereo criterion. Their approach consists in two steps: First an approximative initial estimate of the shape and motion is computed. Then the shape and motion are refined through a multi-resolution spatio-temporal stereo.

More recently, [Pons et al., 2007] proposed a variational method to estimate the shape and the scene flow by maximizing a similarity measure, with respect to shape and motion, between input images and predicted images. A predicted image is the appearance of the surface seen from another camera, or estimated in another time instant using the actual surface, scene flow, and camera parameters. It can be computed by back-projecting an image on the surface and reprojecting it on another camera, which removes simultaneously the projective distortion making the similarity computation much simpler, or by warping an image using the scene flow.

Unlike previous approaches, which compute a matching score on each surface point and integrate it on the surface, they propose to compute the global similarity measure between these images. Consequently, this measure, which is a function that maps two images to a scalar score, can be chosen independently of the approach according to the specific applications, scene properties or imaging conditions. The shape and the scene flow are then computed by minimizing an energy dealing with the similarity term and a regularization term using a gradient descent. This can be implemented easily and efficiently thanks to their matching technique. Their method avoids explicit computation of optical flow, however the shape and motion are still computed sequentially, not taking advantage of the spatio-temporal consistency constraints.

1.2.1.2 Surface tracking / Shape Matching

The scene flow encodes naturally the spatial correspondences between frames. However, due to the finite difference approximations of derivatives, it is usually limited to small displacements between successive frames, high spatio-temporal density is required to estimate correctly the motion field. This is unfortunately not fulfilled in most of real applications.

Another family of approaches estimate the large and fast motions by finding spatio-temporal correspondences, among which some recent ones estimate also dense motion fields by interpolating sparse 3D correspondences. Unfortunately, despite

their high accuracy and their popularity in game and movie industry, marker-based motion capture systems [Menache, 1999, Park and Hodgins, 2006] require very restrictive capturing condition, making then expensive and less practical. This is why researchers are recently more interested in automatic marker-less motion capture techniques. Unlike scene flow estimation, most of these approaches use photometric or/and geometric features to sparsely match the shape at different time instants, i.e. two polyhedral meshes. A dense correspondence field can then be estimated between surfaces using interpolation algorithms.

Doing a complete review of this area is out of the scope of this thesis. We limite ourselves to some of the methods that allow to obtain dense coherence spatio-temporal models from real multi-view video sequences, particularly taking advantage of photometric cues to handle real-world situations robustly. A large number of algorithms have been proposed to solve the 3D correspondence problem geometrically and mathematically. Please refer to [Rusinkiewicz et al., 2005, Gal and Cohen-Or, 2006, Huber and Hebert, 2003, Zhang and Hebert, 1999, Anguelov et al., 2004, Hähnel et al., 2003, Alliez et al., 2007, Bronstein et al., 2007, Hormann et al., 2007] for more information on this area.

[Matsuyama et al., 2004] propose to use photometric and silhouette information to compute shape and motion simultaneously. In their method a surface mesh is deformed under photo-consistency, silhouette and motion flow constraints. Computationally heavy minimization is however required to find dense matches. In [Starck and Hilton, 2005] spherical parameterization is proposed to estimate dense correspondences guaranteeing a continuous bijective mapping between surfaces. Being in a spherical framework, their method is limited to 3D surfaces with genus zero topology and may fail in extreme poses.

More recently, [Ahmed et al., 2008] proposed a method to infer dense 3D correspondences from sparce optical feature-based matches between unrelated adjacent shape-from-silhouette volumes reconstructed for each frame individually. First, SIFT descriptors [Lowe, 1999] are used to find initial coarse correspondences between available reconstructed surfaces, then a scalar monotonic function is associated to each feature making the dense matching possible by comparing the function values on two surfaces. The final estimated correspondences are then used to align a mesh with constant connectivity through all frames. Taking advantage of optical features to find initial sparse correspondences, their algorithm has lower computational cost compared to previous methods, handles arbitrary surfaces, however unlike [Starck and Hilton, 2005] it does not guarantee the one-to-one mapping between surfaces. In an earlier work [Ahmed et al., 2005] they proposed to deform a template human body model by minimizing the reprojection error on silhouettes. This is however

based on pose parameters and can not be applied to general shape and motion.

Some methods propose to evolve a surface under photometric constraints. In [Goldlücke and Magnor, 2004] four-dimensional level-set is employed to evolve a 3D isosurface in space-time. Their method, however has high computational complexity and memory requirement and does not provide 3D correspondences. Silhouettes are also used as constraints for surface deformation. [Starck and Hilton, 2007] combines silhouettes with stereo cues to capture the motion of a surface. More recently, in [de Aguiar et al., 2008] surface- and volume-based deformations are used in a multi-resolution manner to reconstruct the detailed motion and spatio-temporally coherent geometry of performing actor. The scene is represented by a surface mesh and a Delaunay-based tetrahedral mesh, computed initially from a detailed full-body laser scan of the subject registered to the pose at the first frame. Their method consists in two steps. SIFT descriptors, used to find sparse correspondences between consecutive frames, and silhouette information are employed giving a low-detailed global pose estimate in each frame. A volumetric deformation technique based on linear Laplacian deformation is considered increasing the robustness of pose recovery. The surface mesh is then deformed to the estimated global poses, and refined by a multi-view stereo method capturing small-scale surface details. The applied multi-resolution technique makes the method robust to big displacements, capturing in the same time small details. However, a part of high-frequency details is not actually estimated by deformations, but copied from the initial laser-scan geometry. Unfortunately, such precise model is not available in general situations. While applicable to complex shapes and motions, their method does not handle topology changes, and is sensible to the silhouette errors. Deforming laser-scanned models was also employed in their two earlier works [de Aguiar et al., 2007a,b], however because of their flow-based or flow- and feature-based strategy, they were not capable of handling complex motion and shape.

[Varanasi et al., 2008] propose to use both geometric and photometric cues to find sparse, but robust set of matches between successive meshes, estimated initially for each frame by a 3D reconstruction method. SURF [Bay et al., 2006], and normalized geodesic integral [Hilaga et al., 2001] are used as photometric and geometric features respectively. In order to find dense displacement fields, the computed matches are then propagated over all vertices by Laplacian diffusion [Sorkine, 2005], allowing the evolution of the initial mesh between two frames. The resulting mesh is then morphed to the mesh at the second frame guaranteeing the correctness, and its exact overlap with the observation. This is done by the surface evolution approach of [Zaharescu et al., 2007]. Comparing to previous approaches, their method finds sparse matches between large displacements robustly, using both geometric and

photometric features, and handles topological changes and self-intersections through the morphing step. In another recent approach [Furukawa and Ponce, 2008] propose to solve the 3D tracking problem by applying local rigid motion for each vertex and global non-rigid motion on the whole surface of a polyhedral mesh to capture fast and complex motions.

1.3 Discussion and Conclusion

In this chapter, the problems of shape from silhouette and dynamic multiview reconstruction have been reviewed. The most related existing techniques to our work are surveyed, reviewing the different directions followed in the domain. The next chapters describe our novel methods to the problem of multiview reconstruction.

- In Chapter 2, we review some geometric concepts used in our work.
- In Chapter 3, we propose a method to compute a compact and temporally coherent four-dimensional spatio-temporal visual hull of a non-rigid dynamic scene.
- In Chapter 4, we propose a globally optimal multi-view reconstruction method from dynamic cluttered scenes.
- In Chapter 5, we propose a photo-consistent surface reconstruction method based on the geometric concept of medial axis transform.

Chapter 2

Background

In this chapter, some background on the basic computational geometry concepts needed in this thesis is given. More specific concepts will be described in each chapter seperatly. Most of the definitions are taken from [Boissonnat and Yvinec, 1998, Boissonnat and Oudot, 2005]. Please refer to [Boissonnat and Yvinec, 1998, de Berg et al., 1997] for more complete description on this domain.

Contents

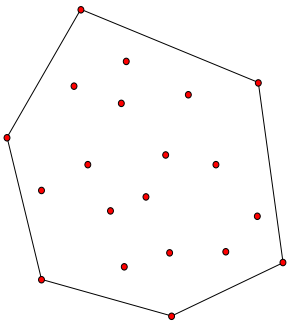


Figure 2.1: Convex hull of a set of points.

2.1 Convex hulls, Polytops, Simplices, and Complexes

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of points in $\mathbb{R}^d$. The *convex hull* of $\mathcal{M}$, $conv(\mathcal{M})$, is the smallest convex set containing $\mathcal{M}$. This is a *polytope*, the convex hull of a *finite* set of points (cf. Figure 2.1 for a two-dimensional example). A plane H is a supporting plane of a polytope $\mathcal{P}$ if it intersects $\mathcal{P}$ and if $\mathcal{P}$ lies completely in one of the two closed half-spaces bounded by H. *Faces* of $\mathcal{P}$ are its intersections with its *supporting planes*.

In the sequel, we call *k-simplex* the convex hull of $k+1$ affinely independent points. For example, a 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. In this work, we also consider 4-simplices: they are known as *pentachorons* or *pentatopes*. Any simplex defined by a subset of vertices of another simplex is a *face* of it.

A *simplicial complex* $\mathcal{C}$ is a finite set of simplices satisfying the following conditions:

1. any face of a simplex in $\mathcal{C}$ is also a simplex in $\mathcal{C}$

2. two simplices in $\mathcal{C}$ either do not intersect, or their intersection is a simplex of smaller dimension which is their common face of maximal dimension.

The *faces* of a simplicial complex are its constituting simplices. In dimension d, the faces of dimension 0, 1, $d-1$, and d are respectively called the *vertices*, *edges*, *facets*, and *cells* of the complex. A k-complex is a complex whose maximal dimension of its simplices is exactly k.

A *cell complex* is defined as a complex whose faces are (possibly unbounded) polytopes. Similar properties should be satisfied:

1. any face of a polytope of $\mathcal{C}$ is also a polytope in $\mathcal{C}$

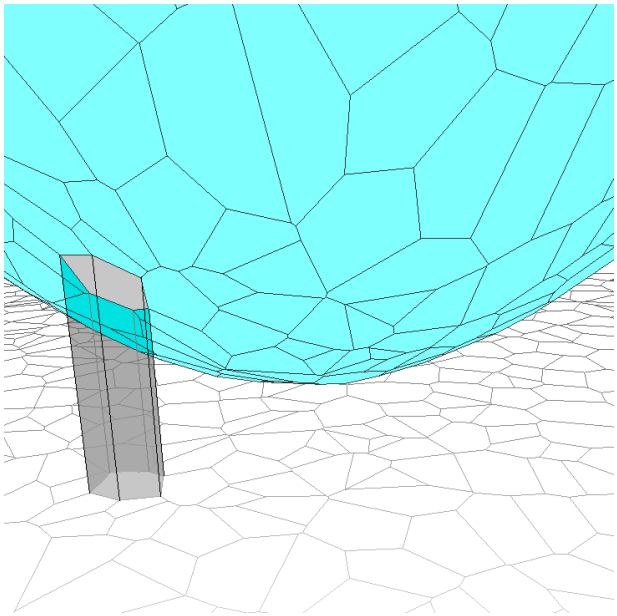


Figure 2.2: The Voronoi polytope and the projected faces on the plane.

2. two polytopes in $\mathcal{C}$ either do not intersect, or their intersection is a polytope of smaller dimension which is their common face of maximal dimension.

Two faces of a complex are *incident* if one contains the other and their dimensions differ by one. Two vertices (cells) of a complex are *adjacent* if they share a common incident edge (facet respectively). According to these relations between faces of a complex, two following graphs are defined:

- *The incident graph* of a complex stores a node for each face of the complex and an arc for each pair of incident faces.
- *The adjacency graph* of a complex stores a node for each cell of the complex and an edge for each pair of adjacent cells.

Two complexes $\mathcal{C}$ and $\mathcal{C}^*$ are called *dual* to each other, if there is a bijection between the faces of $\mathcal{C}$ and those of $\mathcal{C}^*$ which reverses inclusion relationships. A well-known example of such complexes are described in the following sections.

2.2 Voronoi Diagram

Voronoi diagrams are versatile structures which encode proximity relationships between objects. They are particularly relevant to perform nearest neighbor search and motion planning (e.g. in robotics), and to model growth processes (e.g. crystal growth in materials science).

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be set of points in $\mathbb{R}^d$. The *Voronoi region*, or *Voronoi cell*, denoted by $V(p_i)$, associated to a point p_i is the region of space that is closer from p_i than from all other points in E:

$$V(M_i) = \{X \in \mathbb{R}^d \ : \ \forall j, \ \|X - M_i\| \leq \|X - M_j\|\} \ . \tag{2.1}$$

$V(M_i)$ is the intersection of $n-1$ half-spaces bounded by the bisector planes of segments $[M_iM_j]$, $j \neq i$. $V(M_i)$ is therefore a convex polytope, possibly unbounded. The *Voronoi diagram* of $\mathcal{M}$, denoted by $\mathcal{V}or(\mathcal{M})$, is the partition of space induced by the Voronoi cells $V(M_i)$.

See Figure 2.3*(a)* for a two-dimensional example of a Voronoi diagram. In two dimensions, the edges shared by two Voronoi cells are called *Voronoi edges* and the points shared by three Voronoi cells are called *Voronoi vertices*. Similarly, in three dimensions, we term *Voronoi facets, edges* and *vertices* the geometric objects shared by one, two and three Voronoi cells, respectively. The Voronoi diagram is the collection of all these k-dimensional objects, with $0 \leq k \leq d$, which we call *Voronoi objects*. In particular, note that Voronoi cells $V(M_i)$ correspond to d-dimensional Voronoi objects. The Voronoi diagram is a cell complex whose domain is $\mathbb{R}^d$.

Let $\mathcal{P}$ be the paraboloid in $\mathbb{R}^{d+1}$ defined by the equation

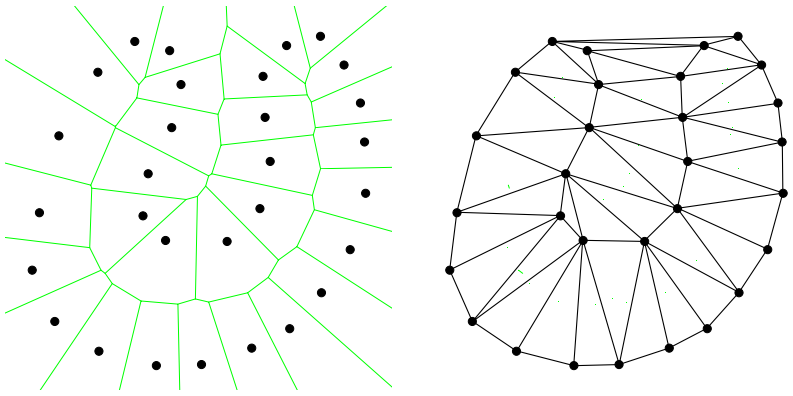
$$x_{d+1} = \sum_{i=1}^{d} x_i^2 \tag{2.2}$$

where $(x_1, x_2, \cdots, x_{d+1})$ are the real coordinates of a point in $\mathbb{R}^{d+1}$. We call the *Voronoi polytope* as the intersection of n half-spaces in $\mathbb{R}^{d+1}$, lying above n hyperplanes tangent to $\mathcal{P}$ at the vertical projections of points of $\mathcal{M}$ onto $\mathcal{P}$ (cf. figure 2.2). It can be shown that the Voronoi diagram of $\mathcal{M}$ can be computed by projecting the faces of the Voronoi polytope onto $\mathbb{R}^d$. The problem of computing the Voronoi diagram can therefore be reduces to the problem of hyper-space intersection.

The following corollary[1] is a result of this theorem. We refer the reader to [Boissonnat and Yvinec, 1998] for the proof and a complete description on the problem of half-space intersection.

Corollary. *The complexity (namely, the number of faces) of the Voronoi diagrams of n points in $\mathbb{E}^d$ is $\Theta\left(n^{\lceil \frac{d}{2} \rceil}\right)$. We may compute such a diagram in time $O\left(n\log n + n^{\lceil \frac{d}{2} \rceil}\right)$, which is optimal in the worst case.*

[1] [Boissonnat and Yvinec, 1998]



(a) (b)

Figure 2.3: *(a)* Voronoi diagram of a set of points in the plane. *(b)* Its dual Delaunay triangulation.

2.3 Delaunay Triangulation

Delaunay complexes are a classical tool in the field of mesh generation and mesh processing due to its optimality properties.

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of points in $\mathbb{R}^d$. Let $\mathcal{P}$ be the paraboloid in $\mathbb{R}^{d+1}$ defined by the equation 2.2, and $\mathcal{D}(\mathcal{M})$ the convex hull of the vertical projection of $\mathcal{M}$ on $\mathcal{P}$ and a point O' on the positive x_{d+1} axis large enough so that the facial structure of the convex hull is stable when O' vanishes to infinity. The Delaunay complex of $\mathcal{M}$, denoted by $\mathcal{D}el(\mathcal{M})$, is obtained by projecting the faces of the lower envelope of $\mathcal{D}(\mathcal{M})$ on $\mathbb{R}^d$. This is dual to the Voronoi diagram: there is a bijection, reversing inclusion relationship, between the k-faces of the Voronoi diagram and the $(d-k)$-faces of the Delaunay complex. When points of E are in L_2-general position[2], $\mathcal{D}el(\mathcal{M})$) is a simplicial complex, which is called the Delaunay triangulation.

The fundamental property of the Delaunay triangulation is called the *empty circle* (resp. *empty sphere* in 3D, resp. *empty hypersphere* in higher dimensions) property: in 2D (resp. in 3D, resp. in 4D), a triangle (resp. tetrahedron, resp. pentatope) belongs to the Delaunay triangulation if and only if its circumcircle (resp. circumsphere, resp. circumscribed hypersphere) does not contain any other points of $\mathcal{M}$ in its interior. The following theorem[3] holds for Delaunay complexes:

Theorem. *Let $\mathcal{M}$ be a set of n points $M_1, \cdots, M_n$ in $\mathbb{E}^d$. Any d-face in the Delaunay complex can be circumscribed by a sphere that passes through all its vertices, and whose interior contains no point in $\mathcal{M}$.*

[2] When no sphere can contain $d+2$ sites on its boundary.

[3] [Boissonnat and Yvinec, 1998]

The computation of the Delaunay complex in $\mathbb{R}^d$ can be reduced to the computation of the convex hull in $\mathbb{R}^{d+1}$. The following corollary[4] is a result of this theorem. Please refer to [Boissonnat and Yvinec, 1998] for a complete description on the computation of the convex hull.

Corollary. *The Delaunay complex of n points in $\mathbb{E}^d$ can be computed in time $O\left(n\log n + n^{\lceil \frac{d}{2} \rceil}\right)$, and this is optimal in the worst case.*

Moreover, as was recently proven in [Attali et al., 2003], the complexity in 3D drops to $O(n \log n)$ when the points are distributed on a smooth surface.

[4] [Boissonnat and Yvinec, 1998]

Chapter 3

Shape from silhouette

Abstract

In this chapter, we propose a novel method for computing a four-dimensional (4D) representation of the spatio-temporal visual hull of a dynamic scene, based on an extension of a recent provably correct Delaunay meshing algorithm. By considering time as an additional dimension, our approach exploits seamlessly the time coherence between different frames to produce a compact and high-quality 4D mesh representation of the visual hull. The 3D visual hull at a given time instant is easily obtained by intersecting this 4D mesh with a temporal plane, thus enabling interpolation of objects' shape between consecutive frames. In addition, our approach offers easy and extensive control over the size and quality of the output mesh as well as over its associated reprojection error. Our numerical experiments demonstrate the effectiveness and flexibility of our approach for generating compact, high-quality, time-coherent visual hull representations from real silhouette image data.

Contents

3.1 Introduction

In this chapter, we propose a novel method to compute a four-dimensional (4D) representation of the spatio-temporal visual hull of a non-rigid dynamic scene. Please refer to 1.1 for an introduction on the problem of shape-from-silhouette and the state of the art in that domain.

Our work builds on a recent provably correct Delaunay-based algorithm for meshing surfaces, from Boissonnat and Oudot [Boissonnat and Oudot, 2005, 2006]. This algorithm is proven to terminate and to construct good-quality meshes, while offering bounds on the approximation accuracy of the original boundary and on the size of the output mesh. The refinement process is controlled by highly customizable quality criteria on triangular facets. A notable feature of this method is that the surface needs only to be known through an *oracle* that, given a line segment, detects whether the segment intersects the surface and, in the affirmative, returns an intersection point. This makes the algorithm useful in a wide variety of contexts and for a large class of surfaces.

The first contribution of our work is to revisit the problem of computing the static visual hull by using the above meshing algorithm. To do so, we have designed a surface oracle and a refinement criterion adapted to multi-view geometry. The resulting algorithm both relates to volume-based (c.f. 1.1.2.1) and surface-based (c.f. 1.1.2.2) approaches.

- Similarly to the volume-based approach, our method builds a decomposition of space, namely the Delaunay triangulation of a set of points sampled from the surface of visual hull. Yet, it is an adaptive unstructured tetrahedral decomposition, in contrast with the usual voxel or octree decomposition, thus eliminating quantization artefacts.

- Similarly to the surface-based approach, our method directly outputs a surface mesh representation, in our case a high-quality watertight triangular mesh. Yet, it is a discrete approximation of the visual hull, under a controlled reprojection error, in contrast with an exact polyhedral intersection, thus avoiding numerical instability and degeneracies in the case of noisy or mutually inconsistent silhouette data.

Moreover, compared to existing static visual hull techniques, our approach has the advantage of offering easy and extensive control over the size and quality of the output mesh as well as over its associated reprojection error.

The second and main contribution of our work is a method to compute a four-dimensional representation of the spatio-temporal visual hull of a non-rigid dy-

namic scene, based on an extension of the meshing algorithm of Boissonnat and Oudot [Boissonnat and Oudot, 2005, 2006] to the four-dimensional space. By considering time as an additional dimension, our approach exploits seamlessly the time coherence between different frames to produce a compact and high-quality four-dimensional mesh representation of the visual hull. The three-dimensional visual hull at a given time instant is easily obtained by intersecting this spatio-temporal mesh with a temporal plane.

Compared to independent frame-by-frame computations, our method has several significant advantages. *First,* it exploits time redundancy to limit the size of the output representation. For example, parts of the scene that are immobile or have a uniform motion can be approximated by a piecewise-linear four-dimensional mesh with few elements elongated in the time direction. We will show in the following that this idea is related to non-uniform meshing depending on the spatio-temporal curvature. In contrast, in the same configuration, a frame-by-frame approach would repeat three-dimensional elements at each frame. *Second,* our method yields a temporally continuous representation, which is defined at any time, thus enabling interpolation of objects' shape between consecutive frames. This also makes a spatio-temporal smoothing of visual hull possible, in order to recover from occasional outliers in silhouette data. *Third,* a byproduct of the two first advantages is the reduction of flicking artefacts in synthesized views, as consecutive three-dimensional slices have a similar geometry and connectivity by construction.

A third contribution of our work is to demonstrate the feasibility of four-dimensional hypersurface representations in computer vision. It is likely to inspire progress in other applications, such as spatio-temporal multi-view stereovision or segmentation of 3D+time MRI sequences of the heart in medical imaging. In that sense, the work of Goldlücke and Magnor [Goldlücke and Magnor, 2004] on spatio-temporal multi-view stereovision is related to ours. These authors take advantage of the fact that the level set method [Osher and Sethian, 1988] easily extends to any number of dimensions. However, their method comes at a very high computation and memory cost, typically requiring several hours if not a day of computation on a cluster to process a few seconds of video. In contrast, despite the fact that our prototype implementation could be heavily optimized, our algorithm builds a compact four-dimensional mesh representation of a few minutes of video in a much shorter computation time on a recent workstation.

The remainder of this chapter is organized as follows. Section 3.2 gives some background on the basic computational geometry concepts needed in our approach. Our novel methods for static and dynamic visual hull reconstruction are described in Section 3.3. In Section 3.4, we report on some numerical experiments which

demonstrate the effectiveness and flexibility of our approach for generating compact, high-quality, time-coherent visual hull representations from real silhouette image data.

3.2 Background

Please refer to chapter 2 for a description on the two geometric concepts needed in our work: The *Voronoi diagram*, and the *Delaunay triangulation* of a set of points. Here we describe the concept of *Restricted Delaunay triangulation*, which we will use in the three- and four-dimensional spaces to approximate the surface of static and dynamic visual hull.

3.2.1 Restricted Delaunay triangulation

Each k-simplex in the Delaunay triangulation is dual to a $(d-k)$-dimensional Voronoi object. In 3D, the dual of a Delaunay cell (a tetrahedron) is the Voronoi vertex which coincides with the circumcenter of the tetrahedron, the dual of a Delaunay facet (a triangle) is a Voronoi edge, the dual of a Delaunay edge is a Voronoi facet, and the dual of a Delaunay vertex M_i is the Voronoi cell $V(M_i)$.

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of points in $\mathbb{R}^d$. Given a subset $\Omega \in \mathbb{R}^d$, typically a manifold of dimension $k \leq d$, we call the *Delaunay triangulation of* $\mathcal{M}$ *restricted to* Ω, and we note $\mathcal{D}el|_\Omega(\mathcal{M})$ the subcomplex of $\mathcal{D}el(\mathcal{M})$ composed of the Delaunay simplices whose dual Voronoi objects intersect Ω. For example, in 2D, as illustrated in Figure 3.1*(b)*, the Delaunay triangulation of a set of points restricted to a curve $\mathcal{S}$ is composed of the Delaunay edges whose dual Voronoi edges intersect $\mathcal{S}$. Similarly, as shown in Figure 3.1*(d)*, the Delaunay triangulation of a set of points restricted to a region Ω is composed of the Delaunay triangles whose circumcenters are contained in Ω. Please see figure 2.3 for the Voronoi diagram and the Delaunay triangulation of the sampled points. The attentive reader may have noticed that in both cases the restricted Delaunay triangulation of a set of points sampled on an object forms a good approximation of the object.

Actually, this is a general property of the restricted Delaunay triangulation. It can be shown that, under some assumptions, and especially if $\mathcal{M}$ is a "sufficiently dense" sample of Ω, (please refer to [Amenta and Bern, 1999] for formal definition), $\mathcal{D}el|_\Omega(\mathcal{M})$ is a good approximation of Ω, both in a topological and in a geometric sense: as regards topology, $\mathcal{D}el|_\Omega(\mathcal{M})$ is homeomorphic to Ω; as regards geometry, the Hausdorff distance between $\mathcal{D}el|_\Omega(\mathcal{M})$ and Ω can be made arbitrarily small; normals and curvatures of Ω can be consistently approximated from $\mathcal{D}el|_\Omega(\mathcal{M})$.

Based on these approximation properties, a family of provably correct algorithms

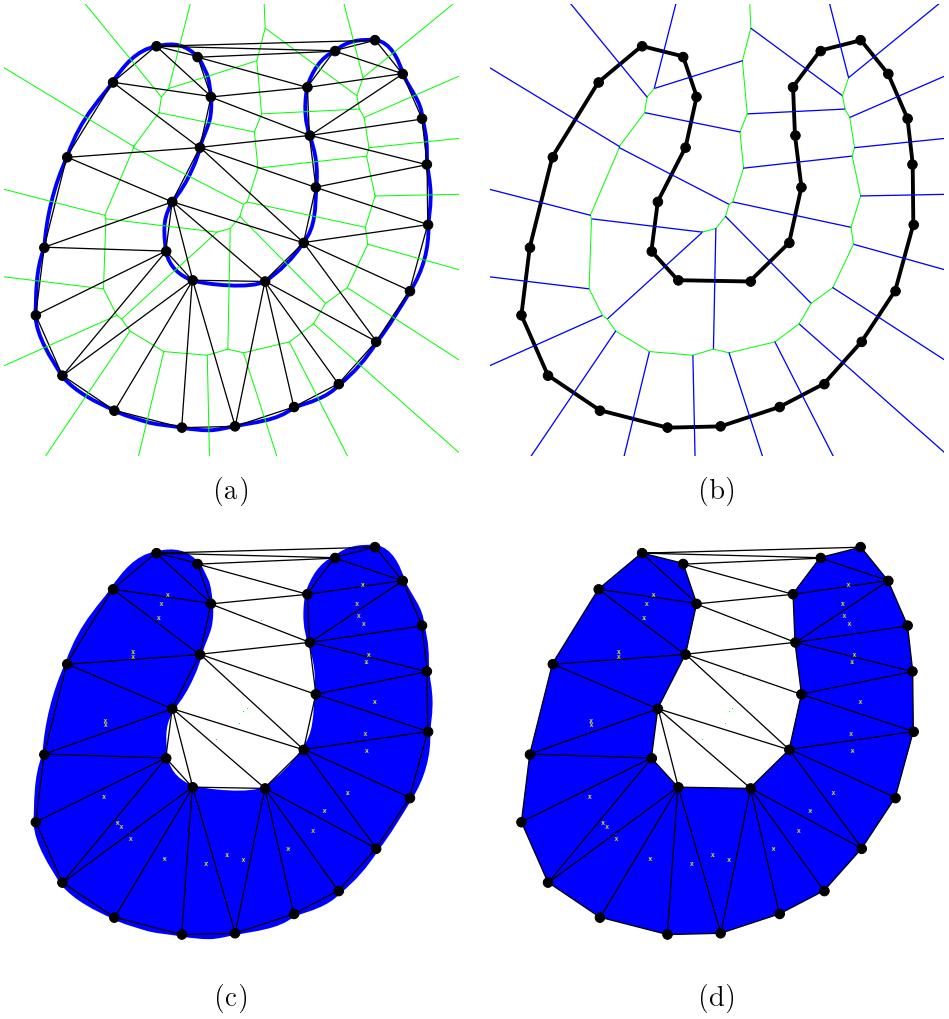


Figure 3.1: *(a)* Voronoi diagram (green) and Delaunay triangulation (black) of a sampled point set restricted to a two-dimensional curve (blue). *(b)* The Delaunay triangulation (black) of the sampled set restricted to the curve approximates the shape of the curve. Only the segments of the Delaunay triangulation whose dual (blue) intersect the curve have been remained in the restricted Delaunay triangulation. *(c)* Delaunay triangulation (black) of a point set sampled from a two-dimensional curve. Duals of the Delaunay triangles which intersect the blue region are shown by crosses. *(d)* The Delaunay triangulation of the sampled set restricted to the region approximates the shape of the region. Only the triangles of the Delaunay triangulation whose dual (crosses) intersect the region have been remained in the restricted Delaunay triangulation.

for mesh generation and mesh reconstruction from point clouds have been designed in the last decade. We refer the reader to [Boissonnat and Oudot, 2005] and references therein for more details.

3.3 Methods

In this section, we first present our novel method for computing the static visual hull, based on the incremental 3D meshing algorithm of Boissonnat and Oudot [Boissonnat and Oudot, 2005, 2006]. Then, we describe in detail the extension of this method to the computation a 4D representation of the spatio-temporal visual hull of a non-rigid dynamic scene.

3.3.1 Static Visual Hull Computation

Let $\mathcal{I} = \{I_1, \cdots, I_n\}$ be the set of input images, and let $\Pi_i : \mathbb{R}^3 \to \mathbb{R}^2$ denote the camera projections from the world reference frame to the image planes. In addition, $\Omega_i \subset \mathbb{R}^2$ are the silhouettes of the object in the different images. The visual hull V is then defined by:

$$V = \left\{ X \in \mathbb{R}^3 \mid \forall i \in \{1, \ldots, n\},\ \Pi_i(X) \in \Omega_i \right\} . \tag{3.1}$$

Let us consider a set $\mathcal{M}$ of points lying on the boundary of the visual hull ∂V. Our approach consists in approximating the visual hull by the Delaunay triangulation of $\mathcal{M}$ restricted to V, *i.e.* in computing the union of tetrahedra of $\mathcal{D}el(\mathcal{M})$ whose circumcenters are contained in the visual hull. The output triangular mesh is then obtained by considering the boundary facets of this set of tetrahedra. Interestingly, this directly enforces watertight surface meshes free of self-intersections.

As mentioned above, it can be proven that this mesh forms a good approximation of the visual hull as soon as $\mathcal{M}$ is a "sufficiently dense" sample of ∂V. With this procedure in hand, our visual hull algorithm reduces to generating a point sample $\mathcal{M}$ which fulfills the above sampling condition as well as some additional user-defined quality and error criteria on boundary facets.

Our algorithm for generating $\mathcal{M}$ closely parallels the surface meshing algorithm of Boissonnat and Oudot [Boissonnat and Oudot, 2005, 2006]. The algorithm starts with a small initial point sample $\mathcal{M}_0$ of ∂V (different strategies to generate the latter are detailed in [Boissonnat and Oudot, 2005, 2006]) and, at each iteration, it inserts a new point of ∂V into $\mathcal{M}$ and updates $\mathcal{D}el|_V(\mathcal{M})$ (please note that all components of the surface must be initially sampled). Each point inserted into $\mathcal{M}$

is the intersection between ∂V and the dual of a boundary facet (that is to say, a ray or a segment of the Voronoi diagram of $\mathcal{M}$). Note that such an intersection always exists, by construction. In case there are several intersections, any of them can be chosen, without compromising the good continuation of the algorithm. The algorithm stops when there are no bad boundary facets left.

The initial computation of $\mathcal{D}el|_V(\mathcal{M})$ and its subsequent updates are fairly simple: it suffices to project the circumcenters of Delaunay tetrahedra in all images and to check whether all the projections lie inside the silhouettes. This check is only performed for tetrahedra that have been modified by the previous point insertion. Similarly, intersections of a segment or a ray with the boundary of the visual hull are computed to the desired accuracy using a dichotomic search along their projection in the different input images.

Under these considerations, the overview of our algorithm is given below:

while there is a bad boundary facet **do**
  let F be the worst boundary facet
  let M be an intersection between ∂V and the dual of F
  insert M in $\mathcal{M}$
  update $\mathcal{D}el|_V(\mathcal{M})$

Figure 3.2 shows a two-dimensional example of the algorithm described above. 3.2(a) shows a two-dimensional curve, initially sampled by three points. The voronoi diagram and the Delaunay triangulation of the points restricted to the curve are computed and shown on the figure. The intersection between the dual of the worst boundary facet (which is a segment or a ray) and the curve is shown by a circle. This point is inserted to the current set of samples, and the Delaunay triangulation restricted to the curve is updated in each step of the algorithm. 3.2(j) shows the restricted Delaunay triangulation after eight steps. This approximates, as expected, the two-dimensional sampled curve.

In the above algorithm, the determination of "good" and "bad" boundary facets is devoted to some user-defined criteria, for example a combination of thresholds on the following elementary quality measures: aspect ratio (minimum angle), size, curvature, edge length, etc. But the most relevant quality criterion in our case is undoubtedly the reprojection error, that is to say the discrepancy between the input silhouettes and the silhouettes of the computed mesh.

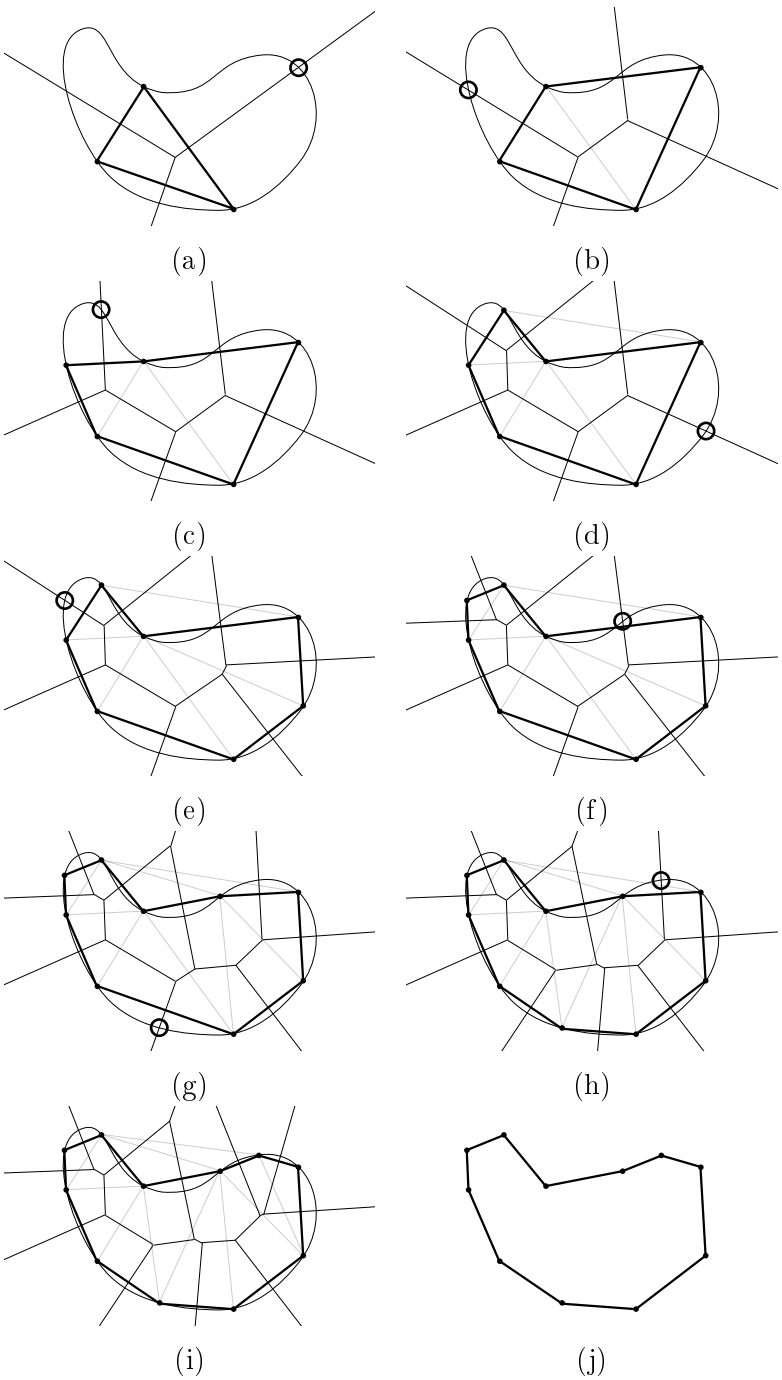


Figure 3.2: A two-dimensional example of our algorithm. A two-dimensional curve, initially sampled by three points, the voronoi diagram, and the Delaunay triangulation of the points restricted to the curve. The intersection between the dual of the worst boundary facet (a segment in 2D) and the curve is shown by a circle. This is inserted to the samples, and the restricted Delaunay triangulation is updated in each step of the algorithm. *(j)* The approximated curve after eight steps.

To this end, we use the signed distance functions $\phi_i : \mathbb{R}^2 \to \mathbb{R}$ associated to the input silhouettes Ω_i. In other words, for a point P in image I_i, we set:

$$\begin{cases} \phi_i(P) = -d(P, \partial\Omega_i) & \text{if} \quad P \in \Omega_i \\ \phi_i(P) = +d(P, \partial\Omega_i) & \text{if} \quad P \notin \Omega_i \end{cases}, \tag{3.2}$$

with $d(P, \partial\Omega_i)$ the distance from the point $P \in I_i$ to the boundary of the silhouette Ω_i. Let us note $\Phi : \mathbb{R}^3 \to \mathbb{R}$ the maximum of the reprojected distance functions,

$$\Phi(X) = \max_i \phi_i \circ \Pi_i(X). \tag{3.3}$$

An alternate definition of the visual hull can be written in terms of the Φ function:

$$V = \left\{ X \in \mathbb{R}^3 \mid \Phi(X) \leq 0 \right\}. \tag{3.4}$$

Interestingly, the maximum reprojection error of a boundary facet with respect to input silhouettes can also be measured with Φ:

$$\text{error}(F) = \max_{X \in F} |\Phi(X)| \,. \tag{3.5}$$

In practice, this error measure is computed by sampling triangular facets and collecting the maximum value of the different ϕ_i at reprojected locations.

During the progress of the algorithm, boundary facets whose error measure exceeds a user-defined threshold (typically one pixel) are further refined. As a result, when the algorithm terminates, the whole output mesh fulfills the desired bound on reprojection error.

3.3.2 Spatio-Temporal Visual Hull Computation

Given multiple video sequences of a moving object and its silhouettes extracted in all images, we could run the above algorithm (or any other static visual hull algorithm) independently in each time frame, and obtain a sequence of three-dimensional visual hull meshes. As discussed previously, this frame-by-frame approach has some

significant deficiencies, because it does not exploit temporal coherence. Here, we propose to construct a "global" spatio-temporal mesh of the sequence, consistent with all input silhouettes.

The main idea of our algorithm is to regard time as a fourth dimension, and to treat it similarly to the three spatial dimensions. At first sight, this is questionable since space is not homogeneous to time regarding physical units. We obtain physical homogeneity of our four-dimensional space by considering a scaling factor v between space and time dimensions. This scaling factor is homogeneous to a speed, and can be interpreted as a reference displacement per time unit, beyond which objects' temporal behavior is not regarded as a continuous motion. So there is no difficulty to tune this parameter for human motion.

We now extend all the definitions of Section 3.3.1 to the dynamic case.

Let $I_i^t,\ i \in \{1,\dots,n\},\ t \in [0,T]$ denote the input video sequences, and $\Omega_i^t \subset \mathbb{R}^2$ the corresponding silhouettes. The scene being captured by fix cameras, the camera projections $\Pi_i : \mathbb{R}^3 \to \mathbb{R}^2$ are constant through time. The spatio-temporal visual hull V is then defined by:

$$V = \left\{(X, vt) \in \mathbb{R}^3 \times [0, vT] \mid \forall i,\ \Pi_i(X) \in \Omega_i^t\right\} . \tag{3.6}$$

There is an important remark we can make here. Actually I_i^t and Ω_i^t are only known at discrete time instants $k\Delta t$, the video frame rate being $1/\Delta t$. A similar remark holds for image locations, but it is likely to be of less practical importance given the increasing resolution of standard consumer video cameras.

Our method for computing an approximating 4D mesh of the spatio-temporal visual hull is a careful extension of the static case. We incrementally build a "good" 4D point sample $\mathcal{M}$ of the boundary of the spatio-temporal visual hull ∂V and we maintain the Delaunay triangulation of $\mathcal{M}$ restricted to V, in other words the union of the *pentatopes* of $\mathcal{D}el(\mathcal{M})$ whose circumcenters lie inside V. It suffices to project the circumcenters of lastly modified pentatopes in the different cameras, and to check whether the projections belong to the silhouettes.

As the fourth coordinate of these intersections is unlikely to correspond to an available time frame, an interpolation of silhouettes between consecutive frames is required. To this end, we use a linear interpolation of the signed distance functions $\phi_i^{k\Delta t}$, which is an established technique in shape statistics [Leventon et al., 2000, Rousson and Paragios, 2002]. As in the static case, we use the signed distance functions $\phi_i^{k\Delta t} : \mathbb{R}^2 \to \mathbb{R}$ associated to the input silhouettes $\Omega_i^{k\Delta t}$ of the frame number k, at the time instant $k\Delta t$. In other words, for a point P in image I_i, at

frame k, we set:

$$\begin{cases} \phi_i^{k\Delta t}(P) = -d(P, \partial\Omega_i^{k\Delta t}) & \text{if} \quad P \in \Omega_i^{k\Delta t} \\ \phi_i^{k\Delta t}(P) = +d(P, \partial\Omega_i^{k\Delta t}) & \text{if} \quad P \notin \Omega_i^{k\Delta t} \end{cases}, \tag{3.7}$$

with $d(P, \partial\Omega_i^{k\Delta t})$ the distance from the point $P \in I_i^{k\Delta t}$ to the boundary of the silhouette $\Omega_i^{k\Delta t}$. The interpolated distance function $\tilde{\phi}_i : \mathbb{R}^2 \times [0, T] \to \mathbb{R}$ is therefore obtained by:

$$\tilde{\phi}_i(P, t) = \phi_i^{k\Delta t}(P) + \left(\frac{t}{\Delta t} - k\right)\left(\phi_i^{(k+1)\Delta t}(P) - \phi_i^{k\Delta t}(P)\right) \tag{3.8}$$

with $k = \lfloor \frac{t}{\Delta t} \rfloor$.

Similarly to the static case, we define also the maximum of the reprojected functions $\tilde{\Phi} : \mathbb{R}^3 \times [0, T] \to \mathbb{R}$ as,

$$\tilde{\Phi}(X, t) = \max_i \tilde{\phi}_i \left(\Pi_i(X), t\right), \tag{3.9}$$

and the reprojection error of a boundary facet (which is a tetrahedra in 4D) with respect to input silhouettes is measures with $\tilde{\Phi}$:

$$\text{error}(F) = \max_{X \in F} |\tilde{\Phi}(X)| \,. \tag{3.10}$$

The output 4D mesh is then obtained by considering the facets of $\mathcal{D}el|_V(\mathcal{M})$. The exact nature of these "facets" deserves clarification: they are tetrahedra with 4D coordinates, so they are indeed *simplicial* pieces of a hypersurface in $\mathbb{R}^4$.

As in the static visual hull algorithm, the construction of $\mathcal{M}$ consists in iteratively adding an intersection point between the spatio-temporal visual hull surface ∂V and the 4D segment/ray dual to the worst boundary tetrahedron of the restricted Delaunay triangulation. In practice, this is computed to the desired accuracy by a dichotomic search along with the aforementioned time interpolation of silhouettes. Again, the refinement process is controlled by a threshold on the reprojection error of boundary facets. This can be done by sampling these tetrahedra and collecting the maximum value of $\tilde{\Phi}$ on the facet. The algorithm terminates as soon as the obtained 4D mesh fulfills the desired reprojection error with respect to all silhouettes of the sequence.

At this point, there are several important remarks to be made. First, the attentive reader may have noticed that the vertices of the 4D output mesh generally do not lie in the input temporal planes $t = k\Delta t$. Second, the space and time density of these spatio-temporal vertices is fully adaptive. Typically, it will be coarse in parts of the visual hull of low curvature and/or of uniform motion. The underlying idea is that regions of ∂V of low *spatio-temporal curvature* can be well approximated with few boundary facets. This is controlled by the quality criterion, allowing to keep the total size of the 4D representation of the scene, and hence the computational and memory cost, sustainable.

Figure 3.3 shows a three-dimensional (2D+time) synthetic example of our spatio-temporal algorithm illustrating the adaptive reconstruction described above. In the case of two-dimensional visual hull, the spatio-temporal representation is a 3D mesh regarding time as the third dimension. In this example, the 2D object is a rectangle which moves vertically in sinusoidal motion, making a 3D sinusoidal object. As seen in the figure, vertices are denser in peaks and valleys of the sinusoid as the temporal curvature is higher in these regions.

Our third remark is intended to overcome doubts about the relevance of interpolating silhouettes between consecutive time frames. These doubts are grounded in the case of a very low frame rate relatively to scene motion. This said, our approach is intended for highest possible frame rates: indeed, our method has the remarkable property that the increase in the number of input frames does not affect either the computational expense or the size of the output. Actually, only complexity of scene geometry and motion matters.

The output 4D mesh cannot be used directly for rendering. Fortunately, the 3D visual hull at a given time instant is easily obtained by intersecting this 4D mesh with a temporal plane. This task can be performed very efficiently, even in real-time on GPUs (Graphics Processor Units), since it reduces to a *marching tetrahedra* algorithm [Gueziec and Hummel, 1995] on the tetrahedra of the 4D mesh, with the temporal coordinate of vertices used as the scalar field for isocontouring. It produces one triangle or one quad per boundary tetrahedron intersected by the selected temporal plane. Note that the produced 3D vertices do not coincide with vertices of the 4D mesh, they are linear interpolations of the latter. Also, their position and connectivity vary continuously with time.

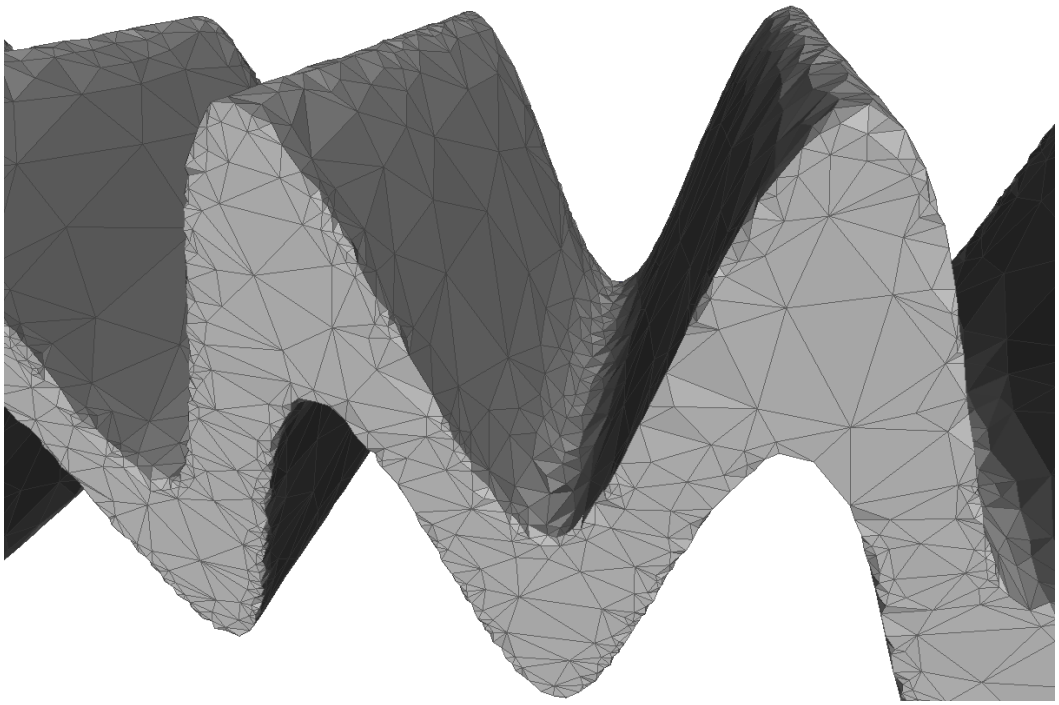


Figure 3.3: A three-dimensional synthetic example of our spatio-temporal algorithm illustrating the adaptive sampling strategy. In the case of two-dimensional visual hull, the spatio-temporal representation is a 3D mesh regarding time as the third dimension. In this example, a rectangle is moving vertically in sinusoidal motion, making a 3D sinusoidal object. As seen in the figure, vertices are denser in peaks and valleys of the sinusoid as the temporal curvature in higher in these regions.

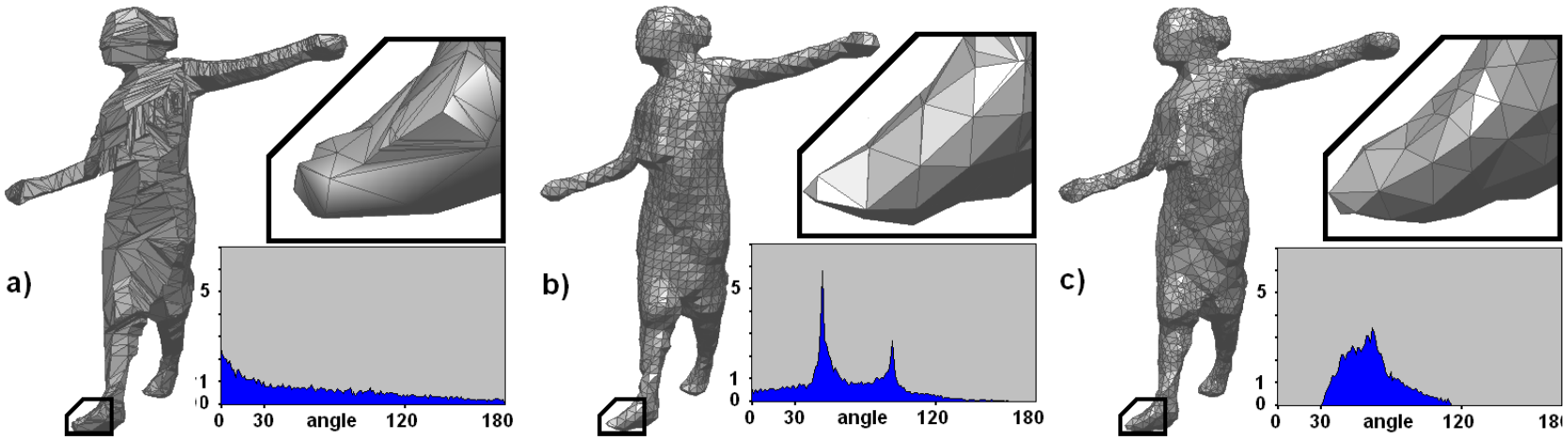


Figure 3.4: Static Visual Hull Reconstruction produced by a) the method introduced in [Franco and Boyer, 2003], b) a volume-based approach followed by marching cube algorithm, c) our method. The magnified views exemplify some typical mesh configurations produced by the three approaches: a) a large number of low quality triangles; b) a uniform resolution mesh containing numerous skinny triangles; c) a non-uniform high-quality mesh. The histograms represent the angle distributions of the obtained surface meshes; the quality of the visual hull meshes is much higher with our approach than with other algorithms.

3.3.3 Implementation Aspects

We have implemented our approach using *CGAL* (Computational Geometry Algorithms Library, homepage: www.cgal.org) [Boissonnat et al., 2000b]. CGAL defines all the needed geometric primitives and provides an excellent algorithm to compute the Delaunay triangulation in 3D: it is robust to degenerate configurations and floating-point error, thanks to the use of exact geometric predicates, while being able to process millions of points per minute on a standard workstation.

CGAL also provides a generic unoptimized Delaunay triangulation algorithm which works in any number of dimensions. We have used this code to implement our spatio-temporal visual hull method. When used in 3D, this generic code runs 20 to 30 times slower than CGAL's dedicated 3D Delaunay code. Consequently, a two orders of magnitude reduction in computation time can be expected in the future, after developing an optimized four-dimensional Delaunay code.

As for the signed distance functions of input silhouettes, we compute them efficiently using *fast marching* [Sethian, 1996].

3.4 Experimental Results

We have tested our algorithms on some real datasets, publicly available at `https://charibdis.inrialpes.fr/`.

3.4.1 Static 3D Visual Hull Reconstruction

Our first experiment focuses on static visual hull reconstruction. We have used the first frame of the "Dance" sequence recorded by eight cameras with resolution 780x582 pixels. In order to illustrate the effectiveness and flexibility of our approach, we compare it to two popular methods, a surface-based approach by Franco and Boyer based on exact polyhedral intersection [Franco and Boyer, 2003], and a volume-based approach followed by a marching cubes algorithm. To provide a fair comparison between the three methods, we adapted the resolution of the 3D image used for the volume-based approach and the refinement criteria of our method (the maximum reprojection error was chosen to be $\max_{X \in F} |\Phi(X)| \leq 0.98$) so that all three meshes would have approximately the same number of vertices. The quantitative results (number of vertices, number of faces, and computation time) are gathered in Table 5.1. Figure 3.5 shows some steps of our reconstruction algorithm, and figure 3.4 displays the reconstructed visual hulls.

In order to show the quality of the output surface meshes, we computed their angle distribution. Figure 3.4 shows the three histograms. Contrarily to our method,

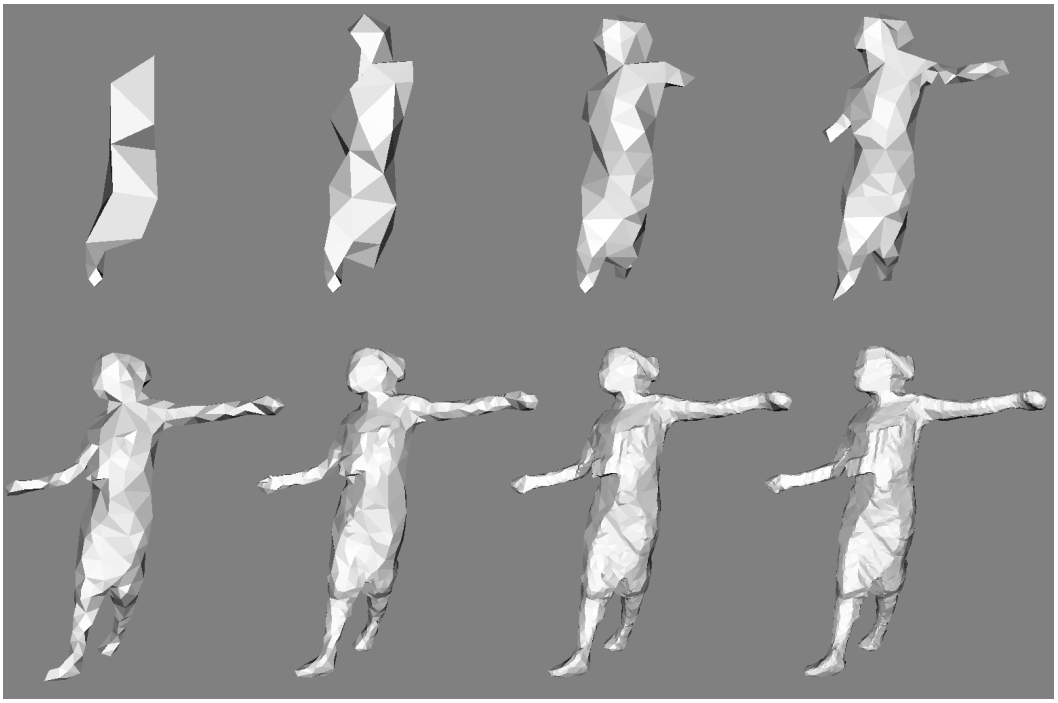


Figure 3.5: Some steps of the reconstruction algorithm for the "Dancer" sequence.

Experiment	# points	# triangles	Time (in sec.)
[Franco and Boyer, 2003]	2400	4790	0.14
Marching cubes	2400	4840	0.04
Our method	2400	4770	2.0

Table 3.1: Parameters and quantitative results of our different numerical experiments.

which produces well-shaped triangles only, other methods yield meshes with lots of skinny triangles.

3.4.2 Spatio-Temporal Visual Hull Computation

In a second experiment, we tested our spatio-temporal visual hull algorithm on a real human motion dataset. We have chosen the "Nicolas2" sequence from the *INRIA Xmas Motion Acquisition Sequences* (IXMAS) dataset. The sequence contains 1084 frames recorded by 5 standard Firewire cameras with resolution 390x291 pixels. We use the silhouette data included in the dataset, which is significantly corrupted by noise and errors, in order to test the robustness of our approach.

Figure 3.6 displays a comparison between our method and the method of Franco and Boyer [Franco and Boyer, 2003], applied independently in each frame. We

Experiment	# points	# points per frame (mean)	Time (minutes)
[Franco and Boyer, 2003]	769356	709	3
Our method	848376	4540 (apparent)	209

Table 3.2: Parameters and quantitative results of our temporal visual hull experiment.

compared the three-dimensional meshes output by their method for some frames of the sequence with the corresponding three-dimensional slices of our four-dimensional spatio-temporal visual hull mesh. To provide a fair comparison, we adapted the spatio-temporal scaling factor v_0 and the refinement criteria of our method (the maximum reprojection error was chosen to be $\max_{X \in F} |\Phi(X)| \leq 1.00$ with a scaling factor $v = 0.72$ (space unit/s)) so that the two methods would construct the whole sequence with approximately the same number of vertices (the number of vertices of our four-dimensional mesh is compared to the total number of vertices of the three-dimensional meshes for the different frames). Table 3.2 shows the quantitative results of the experimented methods.

It appears that, despite an equivalent number of vertices in the output representation, our method yields a better sampled visual hull. This is due to the fact that it generates a much higher number of "apparent vertices" in the three-dimensional time slices than the actual number of four-dimensional vertices. This perfectly illustrates the capability of our approach to take advantage of temporal coherence in order to generate a more compact spatio-temporal representation.

Finally, let us mention that, by computing a four-dimensional representation of the dynamic scene, our method naturally handles topology changes of the visual hull along time, as visible in Figure 3.6 c).

3.5 Discussion and Conclusion

We have proposed a method to compute the spatio-temporal visual hull of a non-rigid dynamic scene, based on four-dimensional Delaunay meshing. Our method outputs a compact and temporally coherent four-dimensional mesh representation of the whole scene. Three-dimensional slices of this mesh can easily be computed for rendering and for interpolating shape between consecutive time frames.

We have validated our approach on real video sequences of human motion. Our results compare favorably with state of the art methods in terms of compactness and quality of the mesh representation. Our work demonstrates the feasibility of four-dimensional hypersurface representations in computer vision. To that extent, we

believe that it can inspire progress in other spatio-temporal problems, such as multi-view stereovision of dynamic scenes, or segmentation of 3D+time MRI sequences of the heart in medical imaging.

A significant drawback of our method is that it is not adapted to the online processing of video sequences. The first reason is its high computational expense. This said, in the future, we expect a two orders of magnitude reduction in computation time thanks to an optimized four-dimensional Delaunay code, thus making interactive processing possible. The second reason is that the very principle of our approach is to treat the video sequence as a whole, thus making offline processing mandatory. Obviously, this approach is not sustainable when the length of the video sequence increases. To overcome this limitation, we consider taking inspiration in the work of Isenburg *et al.* on streaming computation of Delaunay triangulations [Isenburg et al., 2006]. The principle of such an approach would be to restrict computations to a short sliding time window, in order to produce streaming four-dimensional mesh output from streaming video input.

Figure 3.6: Some sample frames of the temporal visual hull reconstruction produced by: a) the method of Franco and Boyer [Franco and Boyer, 2003], b) and c) our method. The magnified views exemplify the significantly higher accuracy offered by our approach, with an equivalent number of vertices.

3.6 Publication

This work has been published in ICCV conferance [Aganj et al., 2007],

- E. Aganj, J.-P. Pons, F. Ségonne and R. Keriven. *Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing.* In IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, Oct 2007.

Chapter 4

Dynamic Multi-view Stereo

Abstract

In this chapter, we propose a method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. Taking visibility into account and being based on a global optimization of a true spatio-temporal energy, it offers several desirable properties: no need for silhouettes, robustness to noise, independent from any initialization, no heuristic force, reduced flickering results, etc. Results on real-world data proves the potential of what is, to our knowledge, the only globally optimal spatio-temporal multi-view reconstruction method.

Contents

4.1 Introduction

In recent years, several methods for automatic generation of complete spatio-temporal models of dynamic scenes from multiple videos have been proposed (cf. Chapter 1). Many of these techniques rely on the visual hull concept [Laurentini, 1994], among which [Aganj et al., 2007, Ahmed et al., 2008, Vlasic et al., 2008]. Computationally efficient, they suffer from several limitations: they provide an approximate reconstruction; this one has to be a closed surface; and, above all, silhouettes have to be segmented in the videos, practically limiting the method to controlled condition capture with a known background. This latest limitation may be lifted when prior knowledge about the geometry is available: free-form deformation of a template body model [Ahmed et al., 2005, Theobalt et al., 2007, Vlasic et al., 2008], Laplacian deformation of a laser scan of the initial pose [de Aguiar et al., 2008, 2007a], etc. Yet, these methods are unable to recover genuine geometric details such as facial expressions and clothing folds and wrinkles. An exception might be the method proposed by Furukawa et al. [Furukawa and Ponce, 2008]. Yielding visually impressive results, this method does not rely on global optimization and handles the occlusion problem via heuristics. Please refer to 1.2 for an introduction on the problem of dynamic reconstruction and the state of the art in the domaine of spatio-temporal reconstruction of non-rigid scenes.

In this chapter, we propose a method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. Taking visibility into account and being based on a global optimization of a true spatio-temporal energy, it offers several desirable properties.

Starting from work by Labatut et al. [Labatut et al., 2007], our method might be seen as its spatio-temporal extension. It is based on modeling an evolving three-dimensional surface as a four-dimensional surface [Aganj et al., 2007, Cheung et al., 2004, Goldlücke and Magnor, 2004]. More precisely, we first generate a quasi-dense 3D point cloud of the scene at each time step and merge the result into a 4D point cloud. This process is conducted in a lenient manner, thus possibly retaining many false matches. Then, we build an adaptive decomposition of the 3D+time space by computing the 4D Delaunay triangulation of this cloud. Finally, we label the Delaunay pentatopes as empty or occupied thus generating a 4D surface. Graph-cut based, this assignment is globally optimal and compatible with the visibility in input images. Optionally but not necessarily, the 3D surfaces corresponding to each time step might be obtained considering 3D slices.

4.1.1 Contributions

Our method has several significant advantages. First, it is not based on visual hulls:

- The videos do not have to be taken under controlled conditions. The background might be cluttered.
- It can handle closed as well as open scenes. For example, it can simultaneously recover the walls and (potentially moving!) furnitures of an indoor scene and a complete reconstruction of subjects seen from all sides in the input images.

Second, it is based on a global optimum:

- It is robust and does not depend on some initialization.
- It exploits visibility information to guide the position of the surface. As a result, it avoids the minimum cut solution from being an empty surface. Hence it exonerates from the usual techniques proposed so far to solve this problem (ballooning term, silhouette information, photo-flux, etc.). Moreover, this visibility information is not enforced as a hard constraint but integrated in the very optimization framework, hence yielding robustness to outliers.

Finally, and mainly, compared to the independent frame-by-frame computations of [Labatut et al., 2007], it profits from the 4D representation:

- Regularization is handled both in space and time, yielding more robustness to noise both in geometry and in visibility reasoning.
- Points extracted at one given time step transfer information to the surrounding time steps. As a result, more details are obtained at each time step.
- Flicking artifacts in synthesized views are reduced, as consecutive 3D slices have similar geometry and connectivity by construction.
- The temporally continuous representation, which is defined at any time, optionally enables interpolation of objects' shape between consecutive frames.

The output of our method might be use for different purposes: as a 4D compact representation; as a list of consecutive 3D meshes; as an initialization for variational spatio-temporal stereovision methods [Goldlücke and Magnor, 2004].

The remainder of this chapter is organized as follows. Section 2 gives some background on the different techniques needed in our approach. In Section 3, we describe in detail the different steps of our algorithm. Section 4 discusses numerical experiments that demonstrate the potential of our approach for reconstructing cluttered scenes from real-world data.

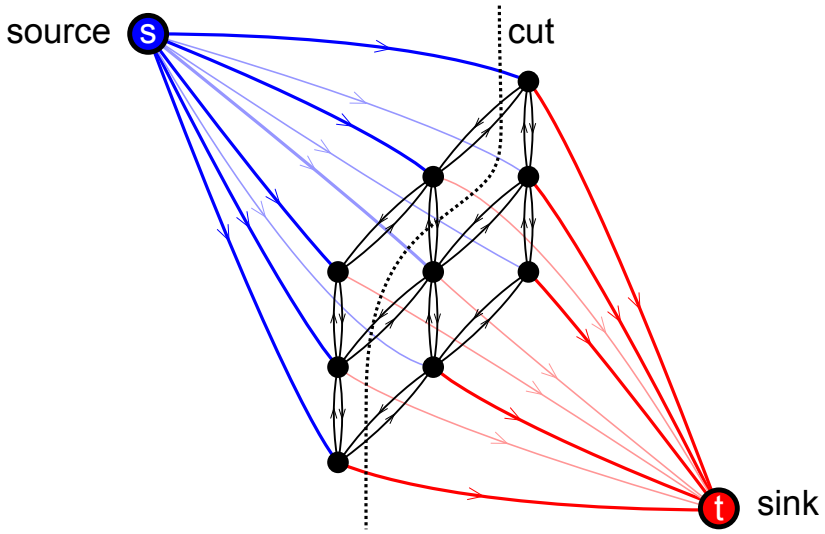


Figure 4.1: A s-t-cut in a graph.

4.2 Background

In this section, we give some background on the minimum s-t-cut problem. We will use this method to obtain the globally optimum spatio-temporal representation of a dynamic scene. Please refer to chapter 2 for a description on the geometric concepts used in our method.

4.2.1 Energy minimization via graph cuts

Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ a directed graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. There are also two *terminal* vertices in $\mathcal{G}$: the source s and the sink t. We assign to each edge connecting p and q a capacity (weight) $c(p, q)$.

A *cut* of $\mathcal{G}$ is a partition of $\mathcal{V}$ into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$. It is called an s-t-cut $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ when $s \in \mathcal{S}$ and $t \in \mathcal{T}$. Figure 4.1 shows an example of a s-t-cut in a graph. A cut-set of $\mathcal{C}$ is the set of edges which go from the source's side to the sink's side, the capacity of $\mathcal{C}$ is defined as the sum of the capacity of all the edges in its cut-set:

$$c(\mathcal{S}, \mathcal{T}) = \sum_{(p,q)\in\mathcal{E}, p\in\mathcal{S}, q\in\mathcal{T}} c_{pq} \tag{4.1}$$

Finding the s-t-cut in a graph which has the smallest capacity is of great interest in many domaines. Indeed, by considering the capacity of a cut as an energy function, the minimum cut computation can be regarded as a binary minimization problem. This is particularly interesting when the energy function is composed of

data terms and *regularization* terms. The first ones represented by the edges between the terminals and the nodes, and the second ones represented by the rest of the edges, not incident to terminals.

It has been shown that this problem is equivalent to the problem of finding the maximum *flow* from s to t (which is intuitively to find the maximum amount of water which can come from the source to the sink, passing through directed edges, not exceeding the capacity of each edge) [Ford and Fulkerson, 1962]: The maximum flow is equal to the minimum s-t-cut. Polynomial-time methods have been proposed to find the globally optimum solution, please refer to [Paragios et al., 2005, Kolmogorov and Zabih, 2004] for an introduction on the graph cuts algorithm and a description on the class of energy functions globally minimizable vie graph cuts.

A large variety of computer vision problems can be formulated in terms of energy minimization. Particularly, many of them reduce to the minimum s-t-cut problem [Greig et al.]. In most of these problems, the image or the 3D space are represented by regular grides. However, more recent works propose to employ the graph cuts algorithm on complexes, globally minimizing energy functions on surfaces defined by the partitions [Kirsanov and Gortler, 2004]. Considering the partitions of a graph as labels on cells of a complex, an oriented surface is identified via a labeling vector: a facet belongs to the surface if it is between an *interior* and an *exterior* cell, naturally oriented according to the labels of its incident cells. Our approach is based on a similar subdivision, employed in a four-dimensional space. Details of our method are described in the next section.

4.3 Method

Our spatio-temporal reconstruction algorithm consists in four steps:

1. A quasi-dense 3D point cloud is generated for each frame, each point memorizing the two or more images from which it has been triangulated. An spatio-temporal 4D point cloud is obtained by adding time as the fourth dimension to all the 3D points;

2. The four-dimensional Delaunay triangulation of the point cloud is computed;

3. The Delaunay pentatopes (4-simplices in $\mathbb{R}^4$) are labeled inside or outside the spatio-temporal surface minimizing some energy, an oriented surface is then extracted as the set of 4D tetrahedra between inside and outside pentatopes;

4. The 3D surface at a given time is obtained by intersecting this 4D hypersurface with a temporal plane.

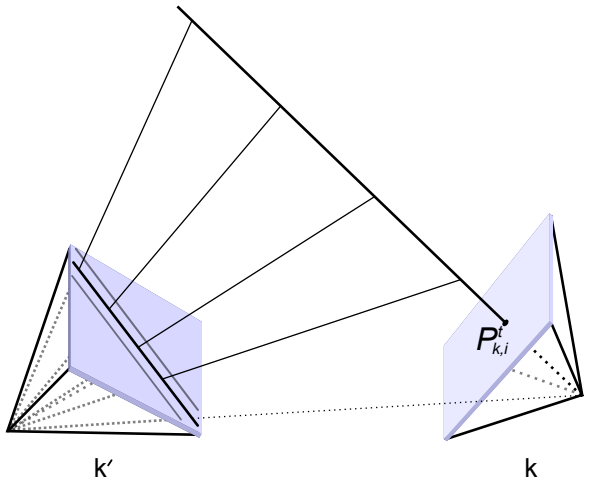


Figure 4.2: A match for an interest point P should lie in a band around the image of the optical ray of P on the other camera.

These steps are described in detail in the rest of this section.

4.3.1 4D point cloud generation, 4D Delaunay triangulation

Given multiple video sequences of a dynamic scene, we first make a dense 3D point cloud for each time instant. Let I^t_k, $k \in \{1, \cdots, n\}$, $t \in [0, T]$ denote the input video sequence: I^t_k is the image captured by camera k at time t. For each image we extract interest points $P^t_{k,i}$ of several types (in practice Harris points and DOGs). For each time instant t, we take all pairs of images $(I^t_k, I^t_{k'})$ and we triangulate the best possible matches $(P^t_{k,i}, P^t_{k',j})$ between their two sets of interest points verifying the epipolar constraint: $P^t_{k',j}$ should lie in a band around the projection of the 3D line passing through the center of camera i and the point $P^t_{k,i}$ on image plane of camera i (Figure 4.2), the width of this band depends on the errors in the point extraction and the calibration processes. We denote $X^t_{kk',ij}$ the 3D point obtained by triangulation of points $P^t_{k,i}$ and $P^t_{k',j}$.

To find the best match for $P^t_{k,i}$, the part of image $I^t_{k'}$ inside a window around the candidate match $P^t_{k',j}$ is reprojected on camera k via a plane passing through $X^t_{kk',ij}$ normal to the optical ray of $P^t_{k,i}$. The matching score is estimated by the normal cross correlation (NCC) between the projected window and corresponding neighborhood of $P^t_{k,i}$. The match is accepted only if their NCC is higher than a threshold, a parameter of the matching process. Finally, in order to take into account all possible points in other images which could match with the reference

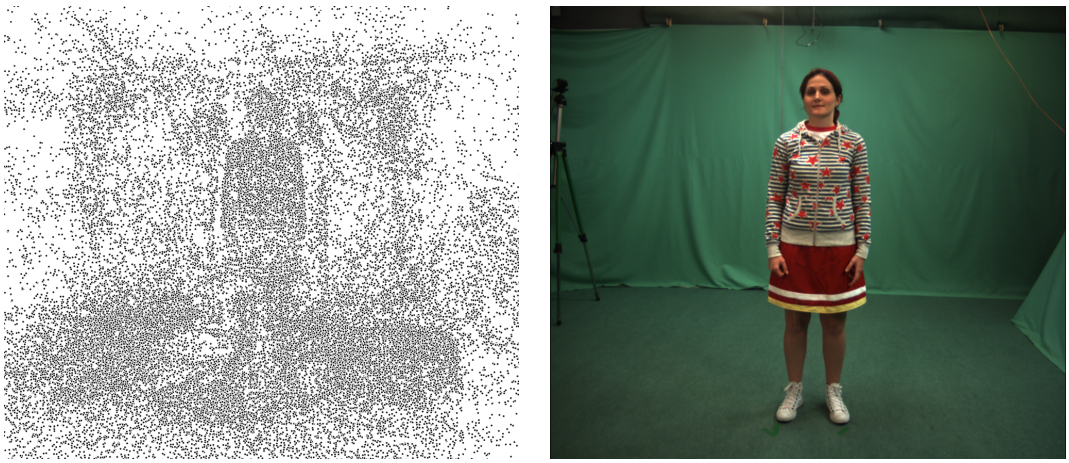


Figure 4.3: The three-dimensional point cloud extracted from 14 images. A large number of outliers appear in the point cloud.

point, we merge the 3D points whose distance is less than a given threshold. Each obtained 3D point is associated to possibly more than two views.

Now we construct a "global" spatio-temporal point cloud by regarding time as a fourth dimension. Similarly to the spatio-temporal 4D space described in chapter 3, we consider a scaling factor v, homogeneous to a speed, between space and time dimensions in order to obtain an physically homogeneous 4D space. The global point cloud is then obtained by taking a 4D point $(X_i^t, vt) \in \mathbb{R}^4$ for each point X_i^t generated from the input images in time t. At the end, we compute the 4D Delaunay triangulation of the spatio-temporal cloud storing in each vertex the list of the views and keypoints from which it has been generated.

4.3.2 4D hyper-surface extraction

In this step we compute a four-dimensional representation of the dynamic scene by extracting a 4D simplicial surface from the Delaunay triangulation computed before. To this end, we label Delaunay pentatopes as inside or outside. The set of facets between differently labeled cells makes a hypersurface, naturally oriented in the four-dimensional space. Actually, a facet of this hypersurface is a tetrahedron oriented from its inside neighbor to the outside one. We note that the normal to a tetrahedron (3-simplex) in $\mathbb{R}^4$ is a four-dimensional vector whose projection on a temporal plane, in our case, gives the spatial normal to the surface.

Let $\mathcal{M} = \{M_1, \cdots, M_n\}$ be the set of n spatio-temporal points in $\mathbb{R}^4$ generated by the method described in section 4.3.1. We denote by $\mathcal{D}el(\mathcal{M})$ the four-dimensional Delaunay complex of $\mathcal{M}$. $\mathcal{D}el(\mathcal{M})$ is the set of all pentatopes, tetrahedra, triangles, segments, and points present in the Delaunay complex. In order

to handle interior scenes, we also add a fictitious *infinite* vertex, with no significant coordinates, to the Delaunay complex. *Infinite pentatopes, tetrahedra, triangles, and segments* incident to this vertex and the convex hull of the points should consequently be added to $\mathcal{D}el(\mathcal{M})$.

We denote by $\mathcal{C} = \{C_1, \cdots, C_m\} \subset \mathcal{D}el(\mathcal{M})$ the set of m cells of $\mathcal{D}el(\mathcal{M})$. We define $\vec{l} = \{l_1, \cdots, l_m\} \in \{I, O\}^m$ a labeling vector which describes the state of Delaunay cells. A cell is labeled I if it is at the interior, or O if it is at the exterior of the spatio-temporal surface of the scene. The later, which we denote by $\mathcal{S}$, is therefore inversely extracted, given a labeling vector $\vec{l}$, as the union of all tetrahedra incident to two differently labeled cells :

$$S(\mathcal{M}, \vec{l}) = \left\{ F \in \mathcal{D}el(\mathcal{M}) \;\middle|\; \begin{array}{l} F \text{ is a facet of } \mathcal{D}el(\mathcal{M}) \\ \exists C_i, C_j \in \mathcal{C}(\mathcal{M}) \text{ such that } \left\{ \begin{array}{l} C_i \cap C_j = F \\ l_i = I \text{ and } l_j = O \end{array} \right. \end{array} \right\} \tag{4.2}$$

Note that the facet F is oriented from the half-space containing the inner cell C_i to the half-space containing the outer cell C_j. The simplicial hypersurface $\mathcal{S}$ is then defined as the union of all facets in S,

$$\mathcal{S} = \bigcup_{F \in S} F \tag{4.3}$$

Our goal is to find the optimal labeling whose associated hypersurface gives the best representation of the dynamic scene. Actually, a small part of these Delaunay cells can already be labeled according to the observation only. A good example is the case of infinite cells: These cells should be labeled as outside when the scene is at exterior, or inside when it is at interior. Figure 4.4 shows the two situations by a two-dimensional example. In left, infinite cells are labeled as outside, the triangle in center in labeled as inside, the surface is therefore oriented to the exterior. In contrast, the infinite cells at righ are labeled as inside, and the triangle at center is labeled as outside. The scene is therefore at interior of the surface, and the faces are oriented to the interior.

Generally, however, such hard constraints are rarely known, the labeling vector should therefore be computed from photometric cues. In the following, we describe how we compute the optimal labeling by minimizing an energy dealing with visibility and spatio-temporal smoothness.

First, we define a neighborhood graph $\mathcal{G} :=< \mathcal{V}, \mathcal{E} >$ between all Delaunay cells $C_i \in \mathcal{C}(\mathcal{M})$. The latters, which are nodes of the graph, are linked by an edge only if

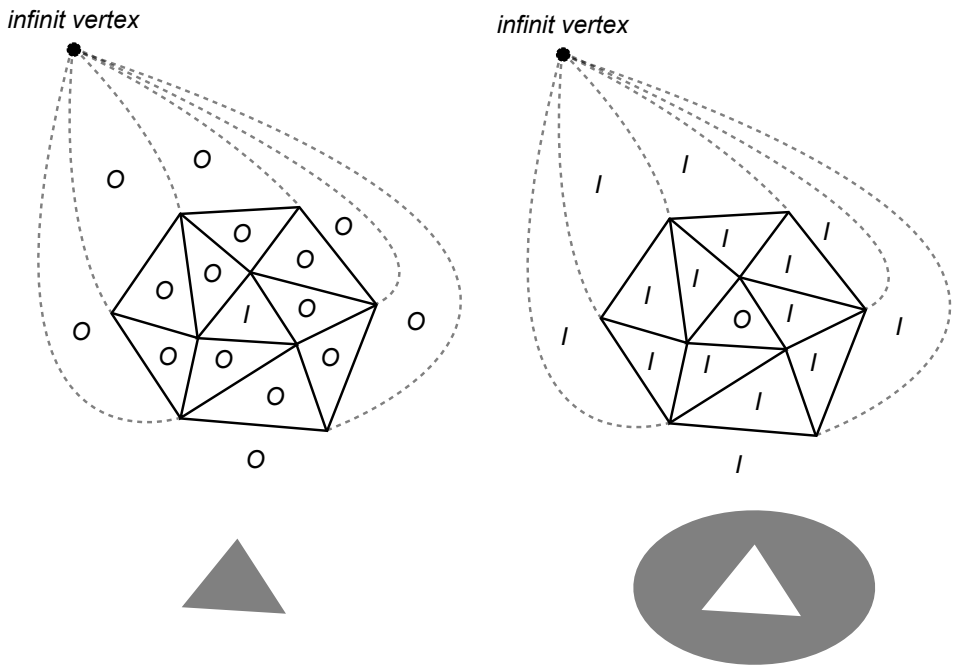


Figure 4.4: A two-dimensional example of the infinite cells. Left: an exterior scene. The infinite cells are labeled as outside. The surface is oriented to the exterior. Right: An interior scene. The infinite cells are labeled as inside. The surface is oriented to the interior.

they share two or three dimensional faces in the Delaunay triangulation, this choice will be discused later. Additionally, all nodes are connected to the sink and to the source:

$$\begin{aligned} \mathcal{G} &:= \quad < \mathcal{V}, \mathcal{E} > \\ \mathcal{V} &:= \quad \{v_1, \cdots, v_m\} \cup \{s, t\} \\ \mathcal{E} &:= \quad \{(s, v_1), \cdots, (s, v_m), (t, v_1), \cdots, (t, v_m)\} \cup \\ &\qquad \{(v_i, v_j) \,|C_i \cap C_j \text{ is a (2 or 3)-simplex}\} \end{aligned} \tag{4.4}$$

Figure 4.5 shows a three-dimensional example[1] of the neighborhood graph constructed from a complex containing four tetrahedra *pqrv*, *stuv*, *pruv*, and *rsuv*. In 3D, only the nodes whose corresponding cells share a one-dimensional, or a two-dimensional face are connected. In this example, the two nodes which correspond to the tetrahedra *pqrv* and *stuv* are not linked in the graph (figure 4.5 right) since they share only a vertex in the triangulation. As described above, all nodes of the graph

[1] For illustration, the example is given in 3D. Indeed, the same 4D arguments can be made in 3D with small modifications.

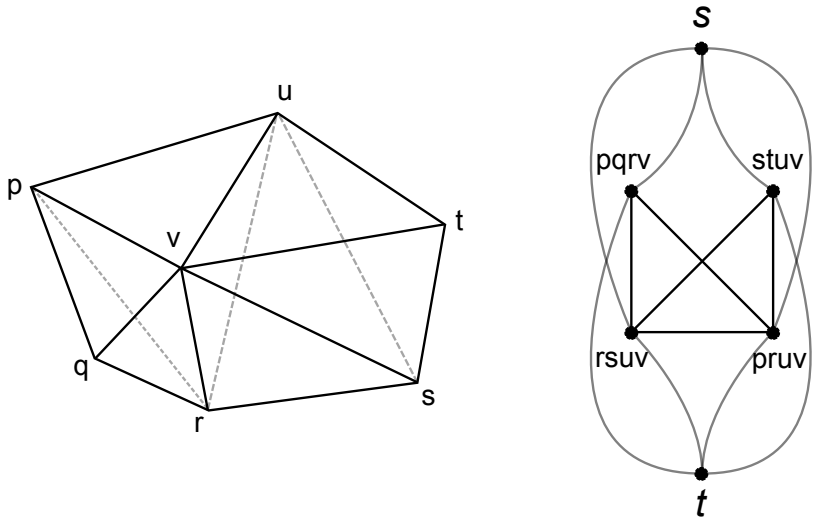


Figure 4.5: A three-dimensional example of the neighborhood graph(right) constructed from a triangulation(left). In 3D, only the vertices whose corresponding cell share a two-dimensional, or three-dimensional face are connected. *pqrv* and *stuv* are not connected, since they share only a point. All vertices are connected to the sourse and the sink.

are also connected to the source and to the sink, here noted by s and t respectively.

Second, according to the energy to be minimized, input images and visibility cue are employed to assign appropriate weights on the edges of the constructed graph $\mathcal{G}$. A globally optimal label assignment is then efficiently found by applying the graph cuts optimization method on $\mathcal{G}$.

In the sequel, we note $\mathcal{S}$ the surface to be reconstructed, which is, as discussed above, a union of four-dimensional tetrahedra. To satisfy both spatial and temporal constraints, we minimize an energy composed of two terms, one dealing with visibility, and the other dealing with spatio-temporal smoothness,

$$\mathrm{E}(\mathcal{S}) = \mathrm{E}_{\text{visibility}}(\mathcal{S}) + \mathrm{E}_{\text{smoothness}}(\mathcal{S}) \tag{4.5}$$

The role of $\mathrm{E}_{\text{visibility}}$ is to penalize the existence of facets which make inconsistent occlusions according to the observation. The idea is that a point can not be an inlier if it is not visible in the camera from which it has been generated. In the rest of this section, we give the exact definition of the energy terms and we describe how they can be implemented in the graph cuts framework.

4.3.2.1 Visibility term

The visibility term that we propose for a spatio-temporal scene is a careful extension of the static case proposed in [Labatut et al., 2007]. The idea of their work is that if a point belongs to the final surface then it should be visible in the views from which it has been triangulated. This yields to the penalization of all the facets intersecting the ray between the point and the cameras from which it has been generated. Additionally, as the point is sampled from the surface, the tetrahedra behind the point should be labeled as inside, encouraging the surface to pass from the point.

In the dynamic case the same argument holds. A point which belongs to the final hypersurface should be visible in all its generating views. Consequently, all 4D pentatope which intersect a 4D ray emanating from the point to the camera center of one of its generating views should be labeled as outside, and the pentatope behind the point should be labeled as inside. We remark that the spatio-temporal center of a camera at a given frame is its 3D center positioned in the temporal plane of that frame. Similarly to the static case, in order to make an energy which can be minimized via graph cuts, we write the visibility term as the number of intersections of a ray, between a point and one of its generating cameras, with the hypersurface, taking into account its correct orientation.

At this point, there are several important remarks to be made. First, the spatial visibility is defined in the 3D space. The visibility of a point at a given frame is therefore defined only in the temporal plane corresponding to that frame. Hence, the rays between the point and its generating views lie completely in the temporal plane which passes through that point.

Second, a 4D facet of the Delaunay triangulation (a 4D tetrahedra) passes generally through several consecutive frames (consider the case where an extracted point cloud is less dense in some parts of it compared to its two neighboring frames. The Delaunay cells would connect its neighbors.). As a consequence, each intersection of a ray with a facet is considered as a penalizing "vote" for the facet. The final vote is then computed as the sum of all votes coming from different frames intersecting the facet. This is an important property since it makes a global visibility vote on every facet taking into account the temporal coherence.

Third, contrarily to the static case presented in [Labatut et al., 2007] where an edge is added to the graph between two Delaunay cells (tetrahedra in 3D) only if they share a facet (triangle in 3D: we remind that in dimension d the simplices of dimension d and $d-1$ are called cells and facets respectively), in the dynamic case, in addition to the cells sharing facets, we add an edge between every two cells (pentatopes in 4D) whose intersection is a 2D face (triangle in 4D).

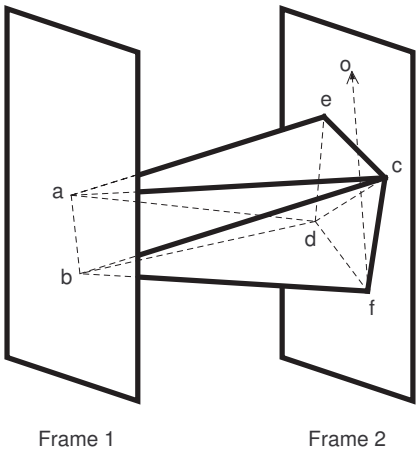


Figure 4.6: The tetrahedra *acde* and *bcdf* have a one-dimensional intersection (the segment *cd*), but they should be connected in the graph (refer to text for details).

Figure 4.6 shows an example of this situation. For simplicity reasons we consider a lower dimensional scene: points are on 2D planes, time is the third dimension and the spatio-temporal object is extracted from the 3D Delaunay triangulation of the point cloud. Points a and b are on frame 1, and points c,d,e and f are on frame 2. Point o is the center of a camera from which f has been generated. The tetrahedra *acde* and *bcdf* have a one-dimensional intersection (the segment *cd*), but they should be connected in the graph since the ray fo intersects *cd* and therefore a penalization term should be added between them. It is important to note that despite the 3D intersection of the ray with the face *acd*, no penalization term should be added between the tetrahedra *abcd* and *bcdf*. That is because *abcd* does not appear in the static representation of the scene on frame 2.

The intersections of the ray with the triangulation can be computed in 4D handling carefully the situation discussed above. However, as a ray always lies completely in a temporal plane, we propose to find these intersections more easily by intersecting the 3D ray with the 3D intersection of the triangulation with the temporal plane, which is computed once for each frame. Obviously only the pentatopes which make a three-dimensional temporal intersection should appear in the temporal slice. In this case, a 3D facet intersected by a ray will correspond to an edge of the graph, and the unnecessary intersections discussed in the example above will be omitted by definition.

We should remark that the 3D intersection of a 4D Delaunay triangulation with a plane contains generally cells with more than four vertices. Figure 4.7 shows the 3D object and a ray intersecting the cells. The nodes p_0, p_1, q_1, p_2 and q_2 shown in

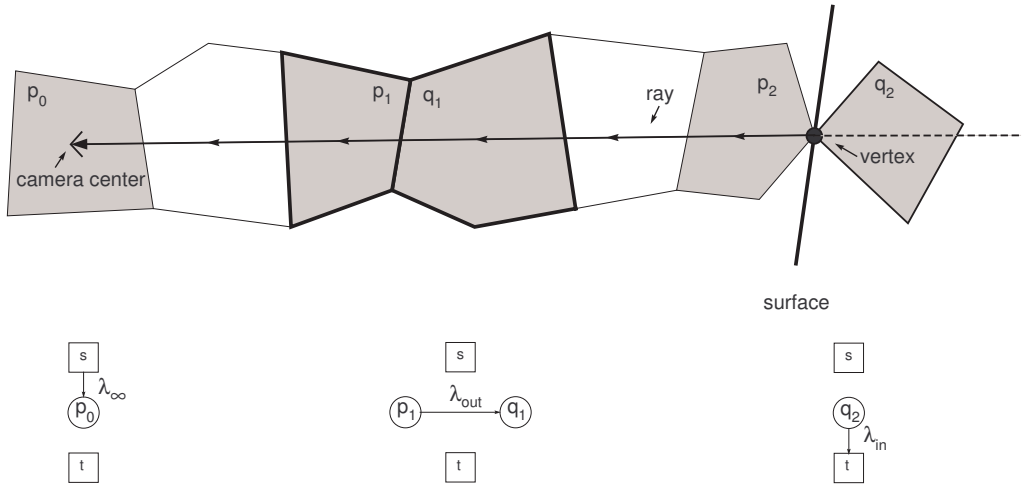


Figure 4.7: Top: A 3D slice of the 4D triangulation. A ray emanating from a vertex to a camera center intersects 3D cells. Bottom: The corresponding visibility-related energy term that penalizes the number of intersections with the ray and the edge weights of the crossed pentatope in the graph.

Figure 4.7(bottom) are the nodes of the graph which correspond to the pentatopes whose temporal intersections make the cells p_0, p_1, q_1, p_2 and q_2 shown in Figure 4.7(top) respectively. Different visibility terms are added. The cell containing the camera should be labeled as outside: a term λ_∞ is added to the edge from source to p_0. A 3D facet crossed by the ray from inside to outside should be penalized: a term λ_{out} is added to the edge from p_1 to q_1. The cell behind the origin of the ray should be labeled as inside: a term λ_{in} is added to the edge from q_2 to the sink.

The positive weights λ_{in}, λ_{out} and λ_∞ take into account the confidence in the reconstructed vertex. By summing up these visibility terms over all the frames, we make a complex distribution of "vote" for each pentatope taking into account the time coherence.

4.3.2.2 Spatio-temporal smoothness

In order to take into account both spatial smoothness and temporal continuity, we propose to minimize the area of the 4D hypersurface in $\mathbb{R}^4$. This yields to the sum of volumes over all the 4D tetrahedra between inside and outside pentatopes,

$$\mathrm{E}_{\text{smoothness}}(\mathcal{S}) = A(\mathcal{S}) = \int_{\mathcal{S}} \mathrm{d}\mathcal{S} = \sum_{F \in S} A(F) \tag{4.6}$$

where $\mathcal{S}$ is the 4D hypersurface to be reconstructed (equations 4.3 and 4.2), F

is a 4D tetrahedra, and $A(F)$ is the volume of the tetrahedra F in $\mathbb{R}^4$. Minimizing this term encourages smoothness in time and in space. As in the static case, this is trivially minimized in the graph cuts framework: for each pair of pentatopes (sharing a tetrahedra F) represented by nodes p and q in the graph, a term $A(F)$ is added the edge $p \to q$ and to its opposite edge $q \to p$.

4.3.3 3D surface extraction

In order to extract a 3D mesh from the 4D output of our method, we intersect it with a temporal plane at a given time instant. This step is similar to the last step of our spatio-temporal shape-from-silhouette algorithm described in Chapter 3. As also mentioned before, this task can be performed efficiently on GPUs (Graphics Processor Units), since it reduces to a *marching tetrahedra* algorithm [Gueziec and Hummel, 1995] on the tetrahedra of the 4D mesh, with the temporal coordinate of vertices used as the scalar field for isocontouring. It produces one triangle or one quad per boundary tetrahedron intersected by the selected temporal plane.

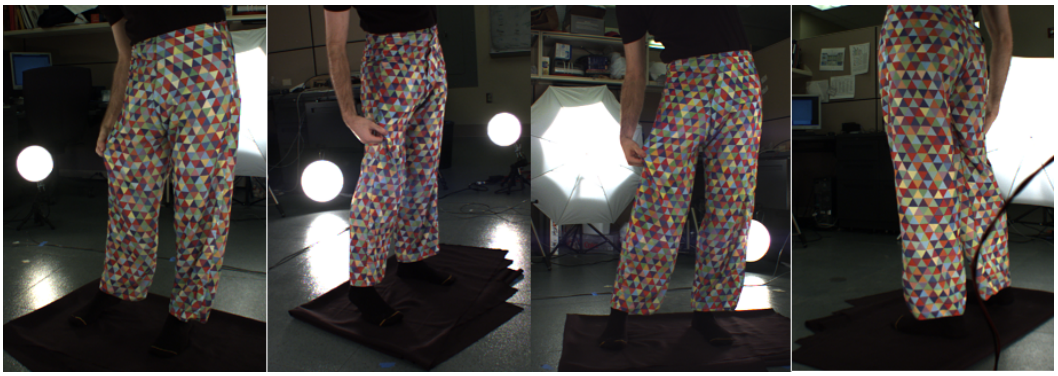


Figure 4.8: Some images of the "Trousers" dataset.

4.4 Experimental results

We have implemented our method using *CGAL* (Computational Geometry Algorithms Library, homepage: www.cgal.org) [Boissonnat et al., 2000b]. CGAL defines most data structure and algorithms needed in our method. For the computation of 4D Delaunay triangulation we have used the Quickhull algorithm library (QHull) [Barber et al., 1996] (homepage: www.qhull.org).

In our first experiment, we have tested our method on the first 60 frames of the "Trousers" sequence which is courtesy of R. White, K. Crane and D.A. Forsyth [White et al., 2007]. The sequence is acquired by 8 cameras at a 640×480 resolu-

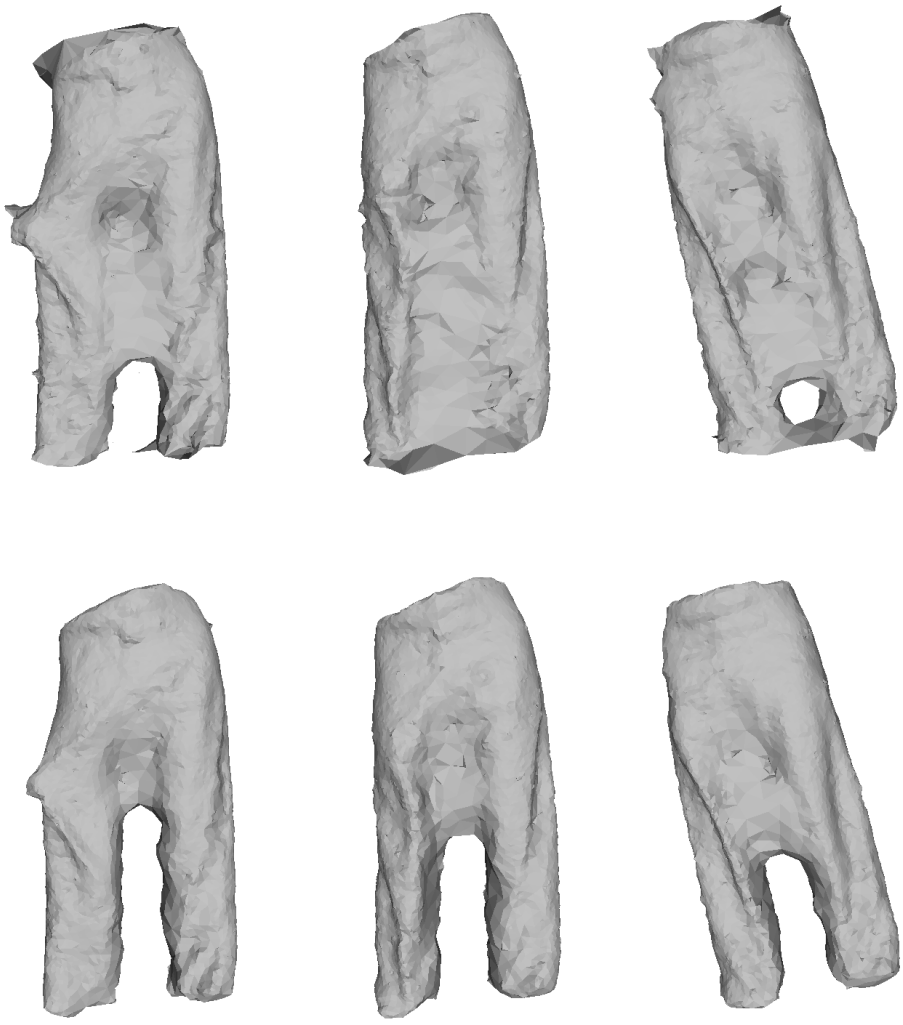


Figure 4.9: A comparison between our method and the method of [Labatut et al., 2007] applied independently in each frame. Top: 3D meshes obtained by the method of [Labatut et al., 2007]. Bottom: corresponding 3D slices of the 4D representation obtained by our method.

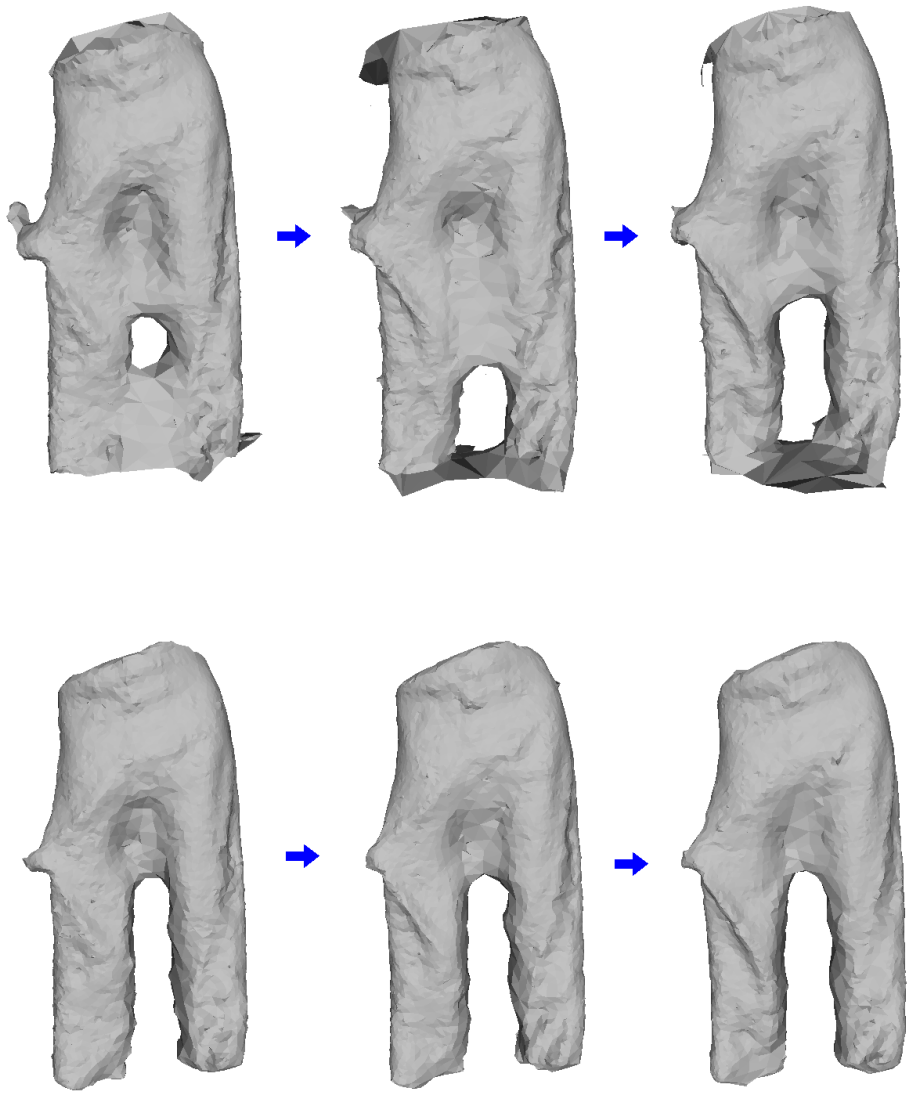


Figure 4.10: The first three consecutive frames of the Trousers dataset reconstructed by (Top): the method of [Labatut et al., 2007] (Bottom): our method. The frame-by-frame reconstruction of the method [Labatut et al., 2007] makes no temporal continuity. In contrast, our method reconstructs correctly the trouser, avoids flicking artifacts and provides a much more continuous motion.

tion. Figure 6.3 shows some images of this dataset. Regarding to an approximate size of the object we have chosen a spatio-temporal scaling factor $v = 30$ (space unit/frame). Figure 4.9 shows a comparison between our method and the method of [Labatut et al., 2007] applied independently in each frame. We have compared the three-dimensional meshes output by their method for some frames of the sequence with the corresponding three-dimensional slices of our spatio-temporal scene. A total number of 793769 points have been generated for the initial point cloud. To provide a fair comparison, we have used the same point clouds in both methods.

We observe that the method of [Labatut et al., 2007] fails to separate correctly the two trouser legs when there is not enough distance between them. In addition, as shown in figure 4.10, their frame-by-frame reconstruction makes no temporal continuity. In contrast, relying on a global optimization, our method reconstructs correctly the trouser, avoids flicking artifacts and provides a much more continuous motion. This perfectly illustrates the capability of our approach to take advantage of temporal coherence in order to obtain more detailed and more continuous result. The computational times of our method and the method of [Labatut et al., 2007] for this experiment are 210 and 112 minutes respectively on a standard workstation. However, the most expensive part of our method is the computation of the four-dimensional Delaunay triangulation. Fortunately, this can be strongly reduced using an optimized four-dimensional Delaunay code.

In a second experiment, we have tested our method on the first 40 frames of the "Dancer" dataset which was made available to us by the 4Dviews company (http://4dviews.com). It is acquired by 14 calibrated and synchronized video cameras. Figure 4.11(top) shows some images of this dataset. The result shows that despite the lowly textured parts of the images, our method makes a correct four-dimensional representation of the dancer. Figure 4.11(bottom) shows some three-dimensional slices extracted from the spatio-temporal object.

Finally, we should remark that in order to have better visualization and to make better comparisons we have smoothed the results of our experiments. However, the output of our method might be used as a four-dimensional compact representation, as a list of consecutive three-dimensional meshes or as an initialization for variational spatio-temporal stereovision methods.

4.5 Discussion and Conclusion

We have presented a new method for multi-view reconstruction from videos adapted to dynamic cluttered scenes under uncontrolled imaging conditions. The main idea of our method is to regard time as the fourth dimension, and to extract a hyper-

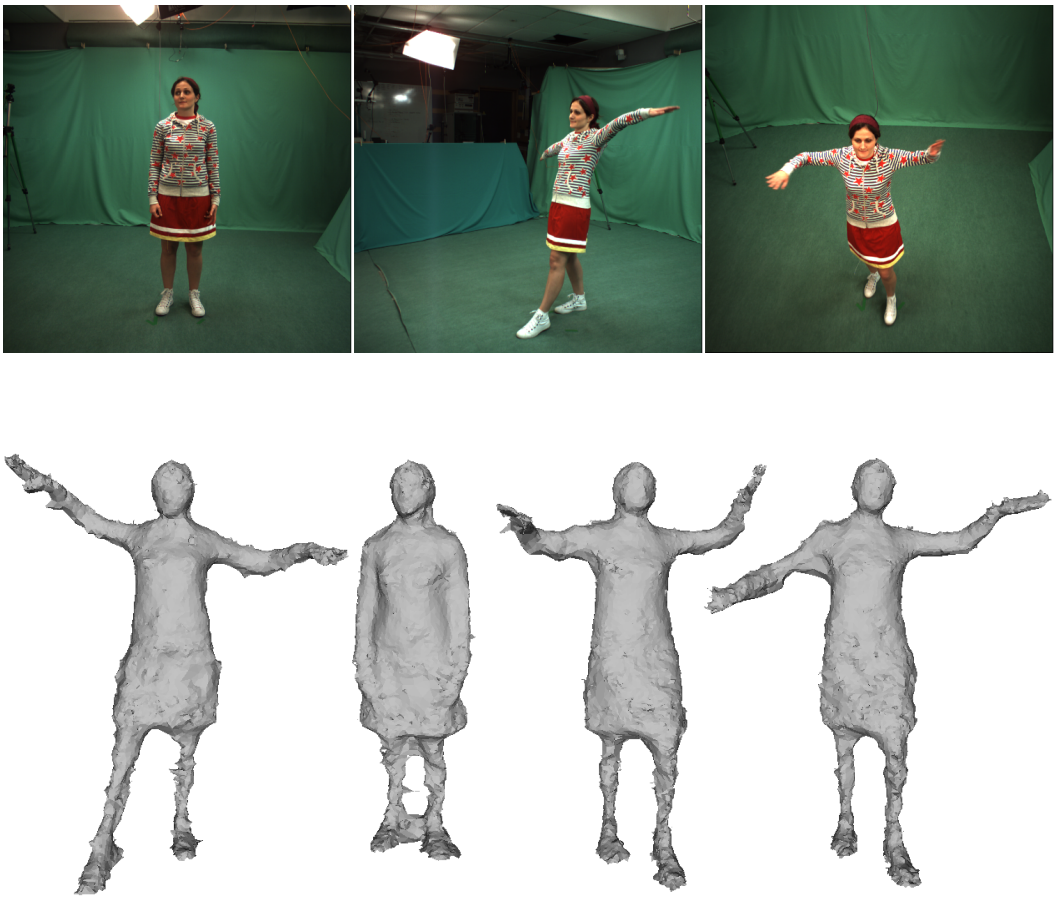


Figure 4.11: Top: Some images of the "Dancer" dataset. Bottom: Some 3D slices extracted from the 4D representation of the "Danser" dataset, obtained by our method. Please note that we do not use the silhouettes or any initialization.

surface from the four-dimensional Delaunay triangulation of the input points as the spatio-temporal representation of the scene. This is done by labeling Delaunay pentatopes as empty or occupied. A globally optimal assignment is efficiently found using graph cuts. We have validated our method on real video sequences. Our results prove the potential of what is, to our knowledge, the only globally optimal spatio-temporal multiview reconstruction method.

4.6 Publication

This work has been published in ACCV conferance [Aganj et al., 2009c],

- E. Aganj, J.-P. Pons and R. Keriven. *Globally optimal spatio-temporal reconstruction from cluttered videos*. In Asian Conference on Computer Vision, 2009.

Chapter 5

Photo-consistent Surface Reconstruction from Noisy Point Clouds

---

Abstract

Existing algorithms for surface reconstruction from point sets are defeated by moderate amounts of noise and outliers, which makes them unapplicable to point clouds originating from multi-view image data. In this chapter, we present a novel method which incorporates the input images in the surface reconstruction process for a better accuracy and robustness. Our approach is based on the medial axis transform of the scene, which our algorithm estimates through a global photo-consistency optimization by simulated annealing. A faithful polyhedral representation of the scene is then obtained by inversion of the medial axis transform. Our work is formulized for the static reconstruction, yet it can be straightforwardly extended to the dynamic case as will be described.

Contents

5.1 Introduction

The problem of approximating a surface from a set of sample points has received a considerable interest in computational geometry, and more generally in mesh processing and in computer graphics. This problem, which is referred to as *surface reconstruction*, arises in many applications of science and engineering. Many approaches have been proposed [Amenta and Bern, 1999, Amenta et al., 2001, Boissonnat and Cazals, 2000], among which some offer theoretical guarantees on the output geometry and topology when the input sampling is sufficiently dense. Of particular interest in this work is the *Power Crust* algorithm [Amenta et al., 2001], which builds a discrete approximation of the medial axis transform, then recovers a polyhedral surface as its inverse.

However, all the aforementioned algorithms assume that the sample is free of noise and outliers. This is an important restriction, since no real scanning device provides exact data. To handle this problem, some recent algorithms have been developed. In [Dey and Goswami, 2006] theoretical guarantees have been provided by considering a noise model in which both the sampling density and the noise level depend on the local level of surface detail. A limitation of their algorithm is that it does not handle arbitrarily over-sampled datasets. This limitation has been overcome in a recent paper [Mederos et al., 2005]. This work augments the *Power Crust* algorithm [Amenta et al., 2001] with a greedy filtering process which discards parts of the medial axis transform originating from noise. However, the noise assumptions used in these two recent works do not hold in practice: the density and the noise level of samples produced by real scanning devices do not depend on the *local feature size* of the surface.

In this work, we tackle the special case of point clouds extracted from calibrated multi-view image datasets. These point clouds typically feature higher levels of noise and higher proportions of outliers than those of active scanning devices. This discards most, if not all, standard surface reconstruction algorithms. Our work continues along the line of [Mederos et al., 2005, Amenta et al., 2001]: it replaces the greedy estimation of the medial axis transform with a global photo-consistency optimization by simulated annealing, thus yielding improved accuracy and robustness.

The remainder of this chapter is organized as follows. Section 5.2 presents some useful computational geometry concepts. Our method is described in Section 5.3 and is experimentally validated in Section 5.4.

5.2 Background

In this section, we give some background on the geometric concepts needed in our algorithm. Please refer to chapter 2 for a description on the *Voronoi diagram* (2.2) and the *Delaunay triangulation* (2.3). We also describe the optimization method of simulated annealing, which we employ to obtain a globally optimal representation of the scene.

5.2.1 Power Diagram and Regular Triangulation

5.2.1.1 Power Diagram

We denote by $\Sigma(C, r)$ a sphere of radius r centered at C in $\mathbb{R}^d$. The *power distance* between two hyperspheres $\Sigma_1(C_1, r_1)$ and $\Sigma_2(C_2, r_2)$ is defined by

$$d^2_{\text{pow}}(\Sigma_1, \Sigma_2) = d^2(C_1, C_2) - r_1^2 - r_2^2 \ . \tag{5.1}$$

Using this distance, we can generalize the Voronoi diagram of a set of points to a set of spheres $\mathcal{S} = \{\Sigma_1, \ldots, \Sigma_n\}$: the *power cell* of a sphere Σ_i is the region of space where a non-weighted point X is closer from Σ_i, in term of power distance, than from all other spheres in $\mathcal{S}$.

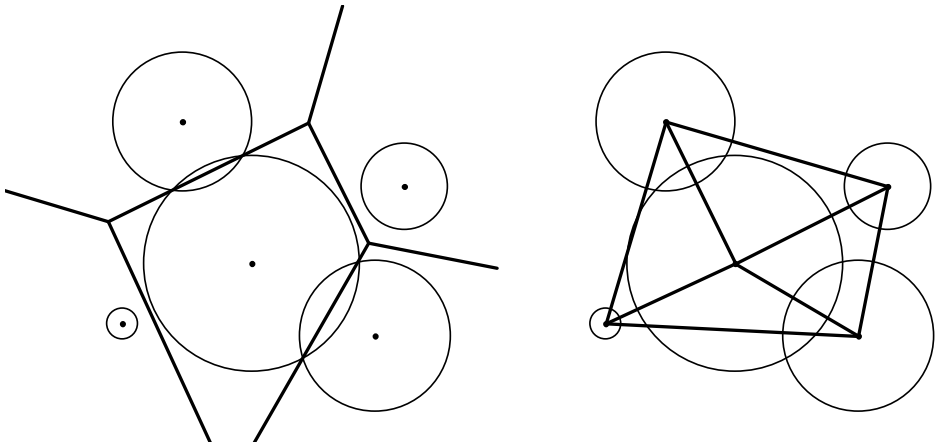
$$P(\Sigma_i) = \{X \in \mathbb{R}^d \ : \ \forall j, \ d_{\text{pow}}(X, \Sigma_i) \leq d_{pow}(X, \Sigma_j)\} \ . \tag{5.2}$$

It can be easily shown that the bisector, in term of power distance, of two spheres is a hyperplane perpendicular to the line passing through their centers. This conicides with the bisector of theirs centers, if they have equal radiuses. The intersection of the half-spaces bounded by the bisector hyperplanes between Σ_i and all Σ_j for $i \neq j$, which contain the points of space with smaller power distance to Σ_i than to Σ_j, is a convex polyhedron, if not empty.

The different power cells of $\mathcal{S}$ induce the *power diagram*, noted $\mathcal{P}ow(\mathcal{S})$, the cell complex containing power regions and their faces. Note that similarly to the Voronoi cells, power cells may be unbounded. The power diagram of a set of spheres coincides with the Voronoi diagram of their centers, if they have equal radiuses. Figure 5.1 (a) shows a two-dimensional example of the power diagram of five circles.

5.2.1.2 Regular Triangulation

By taking the geometric dual of the power diagram, we obtain the weighted Delaunay triangulation of spheres, a *regular triangulation*. There is an edge between the centers of two spheres in the regular triangulation if and only if their associating cells in the power diagram have a non-empty intersection. As in the case of points,



(a) (b)

Figure 5.1: A two-dimensional example of the power diagram (a) and the regular triangulation (b) of five circles.

we obtain a *triangulation* of centers of the spheres in $\mathcal{S}$, a partition of their convex hull into simplices of dimension d. Similarly to the power diagram, the regular triangulation of a set of spheres which have equal radiuses coincides with the Delaunay triangulation of their centers. Figure 5.1(b) shows a two-dimensional example of the regular triangulation of five circles.

Two spheres are orthogonal if their weighted distance is zero. In $\mathbb{R}^d$, $d+1$ spheres have a unique common orthogonal sphere, called power sphere. It can be easily verified that this sphere coincides with the circumsphere of their centers if they are all zero weighted points. The empty sphere property of the Delaunay triangulation extends to the case of power distance. A triangulation of a set of spheres is a regular triangulation if the power sphere of all simplices are *regular*, which is to say their weighted distances to all of the spheres is non-negative.

A sphere can intersect or not its associating power cell. It can also have no power region in the diagram, in which case no cell is associated to it. A sphere with no associated cell does not appear in the regular triangulation.

5.2.2 Medial axis transform

Since its introduction by Blum [Blum, 1967], the *medial axis* (and the *skeleton*, a closely related mathematical notion), has become a standard tool in shape analysis, recognition and classification. Formal definitions vary from author to author. Please refer to a recent review [Attali et al., 2009] for a thorough mathematical presentation.

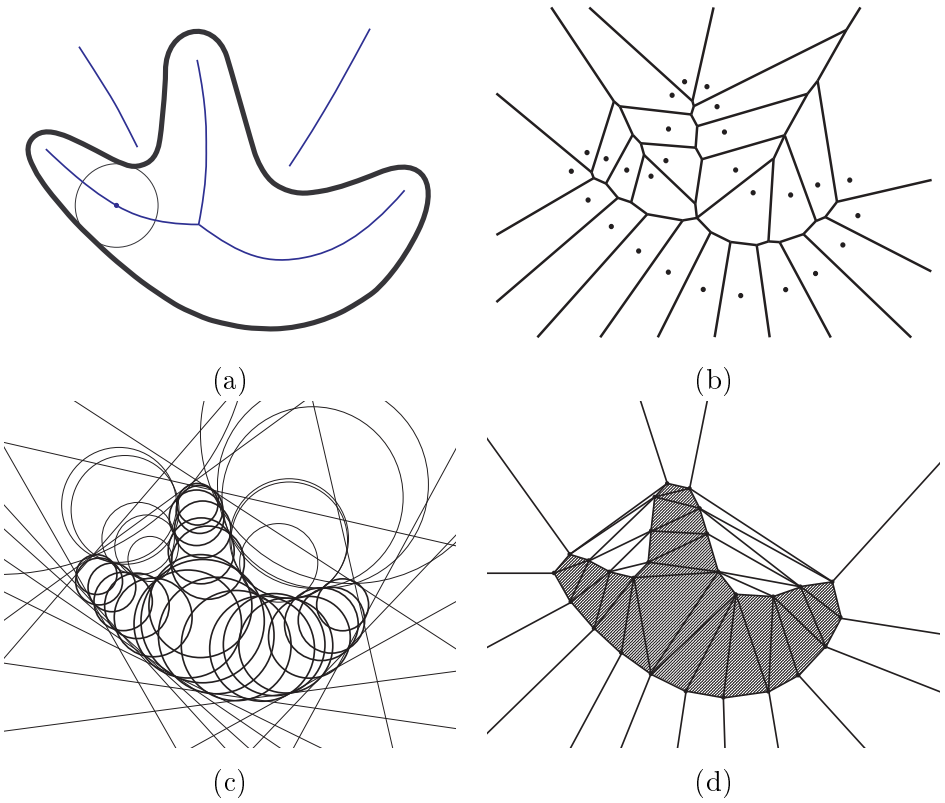


Figure 5.2: *(a)* A two-dimensional curve in plane, its medial axis, and a maximal circle centered in a point of the medial axis. *(b)* The Voronoi diagram of a set of points in the plane. *(c)* The inner and outer polar balls. *(d)* The power diagram of the inner and outer polar balls (see text).

Here, we define the medial axis of a closed bounded surface $\mathcal{S}$ as the closure of the set of points with at least two closest points on $\mathcal{S}$. The *inner medial axis* (resp. the *outer medial axis*) is the subset of the medial axis inside (resp. outside) $\mathcal{S}$. Figure 5.2(a) shows a two-dimensional example of the medial axis of a curve in the plane.

If we weight each point X of the medial axis with the radius $r(X)$ of the maximal sphere centered at X whose interior does not intersect $\mathcal{S}$, i.e. the distance from X to its closest points on $\mathcal{S}$, we obtain the *medial axis transform* of the shape. This transformation is reversible, meaning that $\mathcal{S}$ can unambiguously be reconstructed from its medial axis transform.

5.2.3 Poles and polar balls

We now consider the transposition of the above continuous notions to the discrete case. Of particular interest in surface reconstruction is the result by Amenta

and Bern [Amenta and Bern, 1999] that, given a sufficiently dense sample $\mathcal{M} = \{M_1, \ldots, M_n\} \in \mathbb{R}^3$ of a surface $\mathcal{S}$, the medial axis of $\mathcal{S}$ is approximated by a subset of the *Voronoi diagram* of $\mathcal{M}$.

Following [Amenta and Bern, 1999], we call *poles* the vertices of the Voronoi diagram that approximate the medial axis. A pole is called an *inner pole* or an *outer pole* depending on whether it lies inside or outside $\mathcal{S}$. We call *polar ball* a maximal ball centered at a pole whose interior does not intersect $\mathcal{M}$. In other words, the radius of a polar ball is the distance from the pole to its closest points in $\mathcal{M}$. Figure 5.2(c) shows the set of inner and outer polar balls of the Voronoi diagram 5.2(b).

The set of all inner and outer polar balls is the discrete counterpart of the medial axis transform. Similarly to the medial axis transform, the set of polar balls can be inverted: the *Power Crust* algorithm [Amenta et al., 2001] builds a polyhedral surface that approximates the input point set $\mathcal{M}$, by computing the *power diagram* of the set of polar balls.

Consider the polar balls of a point sample $\mathcal{M}$ of a surface $\mathcal{S}$. The *power crust* is the boundary between the power cells of inner polar balls and the power cells of outer polar balls. It is a polyhedral surface which approximates the surface under some sampling assumptions [Amenta et al., 2001].

5.2.4 Simulated annealing

Simulated annealing (SA) is a probabilistic algorithm invented first by N. Metropolis [Metropolis et al., 1953] for solving combinatorial optimization problems. The concept of this algorithm comes from the annealing in metallurgy where a metal is heated and slowly cooled, aiming to obtain perfect crystallizations. Similarly, the simulated annealing algorithm consists in making random variations on an existing state in a search space, retaining the transformed state if it is better than the original state, and sometimes accepting worse variations with a probability, which is referred as a temperature, that decreases during the process. This technique avoids local minima by randomly jumping to states of higher energy.

The simulated annealing is usually employed in discrete optimization problems, searching for a good approximation of the global minimum of a function. Indeed, it can be shown that the probability of reaching the global optimum approaches 1 [Granville et al., 1994]. The algorithm starts with an initial state s_0, and an initial temperature T_0 which will decrease during the process. In each iteration, a neighbor state s_n is randomly chosen, and the change in energy δE is computed. If δE is negative, the system moves to the new state s_n. Otherwise, it decides with a probability, a function of δE and T which is usually the Boltzmann factor $e^{-\frac{\delta E}{T}}$, if

it should still go to the new state. This process is repeated, while the temperature is decremented, until a sufficiently good energy or a sufficiently low temperature is obtained, or a defined number of iterations has passes.

5.3 Approach

We now focus on the problem of surface reconstruction: $\mathcal{S}$ is unknown. $\mathcal{M}$ may not fulfill the aforementioned sampling assumptions, and is corrupted by a certain amount of noise and outliers. The previous notions are then faced to significant difficulties: (i) the classification of poles as inner or outer is no more straightforward, (ii) due to noise and outliers, some poles may not approximate the medial axis.

In the original *Power Crust* algorithm [Amenta et al., 2001], poles are labeled as inner or outer by a greedy propagation process driven by a priority queue. Unfortunately, this greedy approach has been shown to fail with a moderate amount of noise and outliers in [Kolluri et al., 2004].

In our work we propose to replace the greedy estimation of the medial axis transform with a global photo-consistency optimization by simulated annealing. A faithful polyhedral representation of the scene is then obtained by inversion of the medial axis transform.

5.3.1 Formulation

Given a sample set $\mathcal{M} = \{M_1, \ldots, M_n\} \in \mathbb{R}^3$, we denote by $\mathcal{B} = \{B_1, \cdots, B_m\}$ the set of m polar balls of the corresponding Voronoi diagram. We define $\vec{l} = \{l_1, \cdots, l_m\} \in \{-1, 0, +1\}^m$ a labeling vector describing the state of the poles: a pole is labeled -1 if it is at the interior, or $+1$ if it is at the exterior of the surface. A pole labeled 0 is considered as noise and will be discarded in the estimation of the medial axis transform.

We denote by $\mathcal{S}_\mathcal{B}(\vec{l})$ the inversion of the medial axis transform estimated by the set of labeled polar balls $(\mathcal{B}, \vec{l})$. This is a polyhedral surface obtained by the power crust of the poles in $\mathcal{B}$ labeled inside (-1) or outside $(+1)$ as described before.

In order to estimate correctly the medial axis transform we wish to minimize an energy functional dealing with photo-consistency on the surface of power crust,

$$\mathrm{E}(\mathcal{S}_\mathcal{B}(\vec{l})) = \sum_{F \in \mathcal{S}_\mathcal{B}(\vec{l})} \gamma(F) \tag{5.3}$$

where F is a face of the polyhedral surface $\mathcal{S}_\mathcal{B}(\vec{l})$. This measures how well the given surface matches the different input images in which it is seen. It is defined as the sum over the whole surface of some photo-consistency measure γ, which is in

practice computed by the normalized cross-correlation of images reprojected on each other via the surface. A face of $\mathcal{S}$ is oriented, since it is the Voronoi face between an outer and an inner pole. Hence, the photo-consistency of each face is computed only in the views which make small angle with the normal to the face. The optimal surface is obtained by

$$\mathcal{S}_{\text{opt}} = \mathcal{S}_{\mathcal{B}} \left(\arg\min_{\vec{l}} \mathrm{E}(\mathcal{S}_{\mathcal{B}}(\vec{l})) \right) \tag{5.4}$$

In the next section we describe why the simulated annealing can be used efficiently for the optimization process. We give then the detailed algorithm to find the opimal surface.

5.3.2 Energy Variation

As the simulated annealing algorithm searches the optimum by making small modifications on the state of the system in an iterative manner, it is important to show that the modifications of the energy function can be computed efficiently in each iteration. To this end, we consider the variation of energy with respect to its variables: the poles.

The energy function 5.3 is the sum of photo-consistency/smoothness terms over Voronoi faces of the power diagram which are between outer and inner poles. The energy term of each face can be computed independently of the others. Hence, to compute the variation of the energy function, we should consider the modification of the faces when poles modified.

When $\vec{l}$ is modified:

- some faces are removed (F_r), added (F_a), or partially modified (F_m)
- some faces are not modified

As the energy is integrated over all faces of the surface, its variation can be computed by

$$\begin{aligned} \Delta E &= E(F_a) - E(F_r) + \Delta E(F_m) \\ |\Delta E| &= |F_r \cup F_a \cup F_m| \end{aligned} \tag{5.5}$$

where $|\Delta E|$ is the number of terms to be computed. Modifying (inserting, removing or relabeling) a polar ball B_i in the power diagram affects only Voronoi faces with at least one common vertex with the corresponding cell of B_i. This consists of all faces of the cell of B_i, and a subset of faces of it's neighbor cells. Hence, $|\Delta E|$ is limited by the number of neighbor faces of B_i. The energy variation can be therefore computed efficiently by considering only the faces mentioned above.

5.3.3 Algorithm

In order to minimize 5.3, we use the simulated annealing optimization algorithm [Metropolis et al., 1953]. The algorithm starts with an initial labeling vector $\vec{l_0}$. A global variable T is taken as the temperature of the optimization process, it is initialized to T_0. At each iteration, it takes a random pole B_i in $\mathcal{B}$. Then, it changes its label l_i to a random new state and computes the energy variation δE. If δE is negative, it accepts the new labeling and goes to next iteration. Otherwise, it decides with a probability given by a function $\Pr(\delta E, T)$ if the new state should be accepted. In case it should not, it returns to the old configuration. The temperature T decreases during the process by a function which depends on the number of passed iterations. The algorithm stops when the energy is sufficiently low, or when a predefined time limit is over. The algorithm returns $\mathcal{S}_{\mathcal{B}}(\vec{l})$ as the final surface.

The overview of our algorithm is given below:

$\vec{l} \Leftarrow \vec{l_0}$
$T \Leftarrow T_0$
while $E > E_{\min}$ **or** time $>$ time$_{\max}$ **do**
  $B_i \Leftarrow$ RandomPole[a]$(\mathcal{B})$
  $\vec{l}_{\text{old}} \Leftarrow \vec{l}$
  $l_i \Leftarrow$ RandomState[b]$()$
  $\delta E \Leftarrow E(\vec{l}) - E(\vec{l}_{\text{old}})$
  if $\delta E > 0$ **then**
    if $\Pr(\delta E, T) <$random[c]$()$ **then**
      $\vec{l} \Leftarrow \vec{l}_{\text{old}}$
  $T \Leftarrow$ decrease(T)

[a] The call RandomPole($\mathcal{B}$) returns randomly chosen pole among the poles of $\mathcal{B}$.
[b] The call RandomState() returns a random state: $(-1, 0, +1)$.
[c] Then call random() returns a random value in the range $[0, 1]$.

5.4 Experimental Results

We have tested our method on a real dataset publicly available at `http://cvlab.epfl.ch/~strecha/multiview/`. All implementations have been done using *CGAL* library (`http://www.cgal.org/`)[Boissonnat et al., 2000a].

We have compared our method with a recent multi-view reconstruction method [Labatut et al., 2007] robust to outliers. Our experiment uses the eight views Herz-

Jesu-P8 dataset from [Strecha et al., 2008]. Some images of this dataset are shown in Figure 5.5(c). 6950 points are generated by matching image keypoints. In order to reduce the computation time of our method we have initialized it with a labeling vector obtained by [Labatut et al., 2007] and we have reduced the optimization space to the set of polar balls with radius smaller than the characteristic noise level of the dataset. The optimization process terminated after 40 minutes on a standard workstation. Figure 5.3 shows the evolution of the energy function during the simulated annealing process.

Two other methods have also been tested as sanity checks: the reconstruction algorithm [Mederos et al., 2005] and the Laplacian smoothing algorithm which have been both initialized by the result of [Labatut et al., 2007]. We have compared the cumulative error distribution of the four meshes with respect to the mesh obtained by the LIDAR technique provided by [Strecha et al., 2008]. Table 5.1 shows the error distribution of the results provided by an automatic multi-view evaluation program [Strecha et al., 2008]. For each method four cumulative error distributions have been computed. Column n represents the percentage of image pixels with an error less than $n\sigma$, where σ is the characteristic noise of LIDAR. More accurate results yield to higher scores in the table. Figure 5.4 shows the meshes obtained by four experimented methods. Our result compares favorably with the three other methods in terms of accuracy and robustness.

In a second experiment we have initialized a variational multi-view stereovision method [Pons et al., 2007] with our mesh and with the output of [Labatut et al., 2007]. Figures 5.5(a) and 5.5(b) show the results obtained by the two initializations. The results show that [Pons et al., 2007] provides more accurate and less noisy surface when it is initialized by our method.

Experiment	1	3	5	7
[Labatut et al., 2007]	2.37	10.81	16.80	20.89
Laplacian Smooth	2.21	10.18	15.80	20.14
[Mederos et al., 2005]	2.38	10.84	16.87	20.98
Our method	3.60	15.10	21.79	25.72

Table 5.1: Quantitative results of our different numerical experiments. Column n represents the percentage of image pixels with an error less than $n\sigma$. More accurate result yields to a higher score in the table.

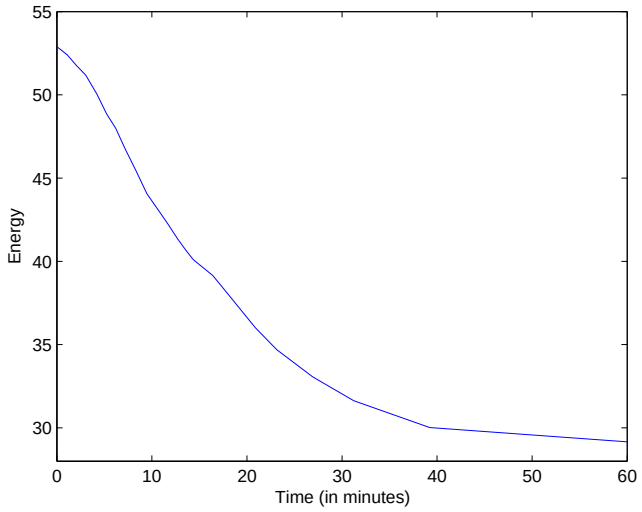


Figure 5.3: The evolution of the energy function during the simulated annealing process.

5.5 Discussion and Conclusion

We have proposed a photo-consistent surface reconstruction method from noisy point clouds based on the estimation of the medial axis transform. Our work replaces the greedy algorithms of [Mederos et al., 2005, Amenta et al., 2001] with a global photo-consistency optimization by simulated annealing. We have validated our method on real datasets. Our results compare favorably with state of the art methods in terms of accuracy and robustness to noise. A drawback of our method is its computational complexity due to the optimization process. We hope that by modifying the form of the energy function we will be able to use more practical optimization algorithms such as graph-cuts.

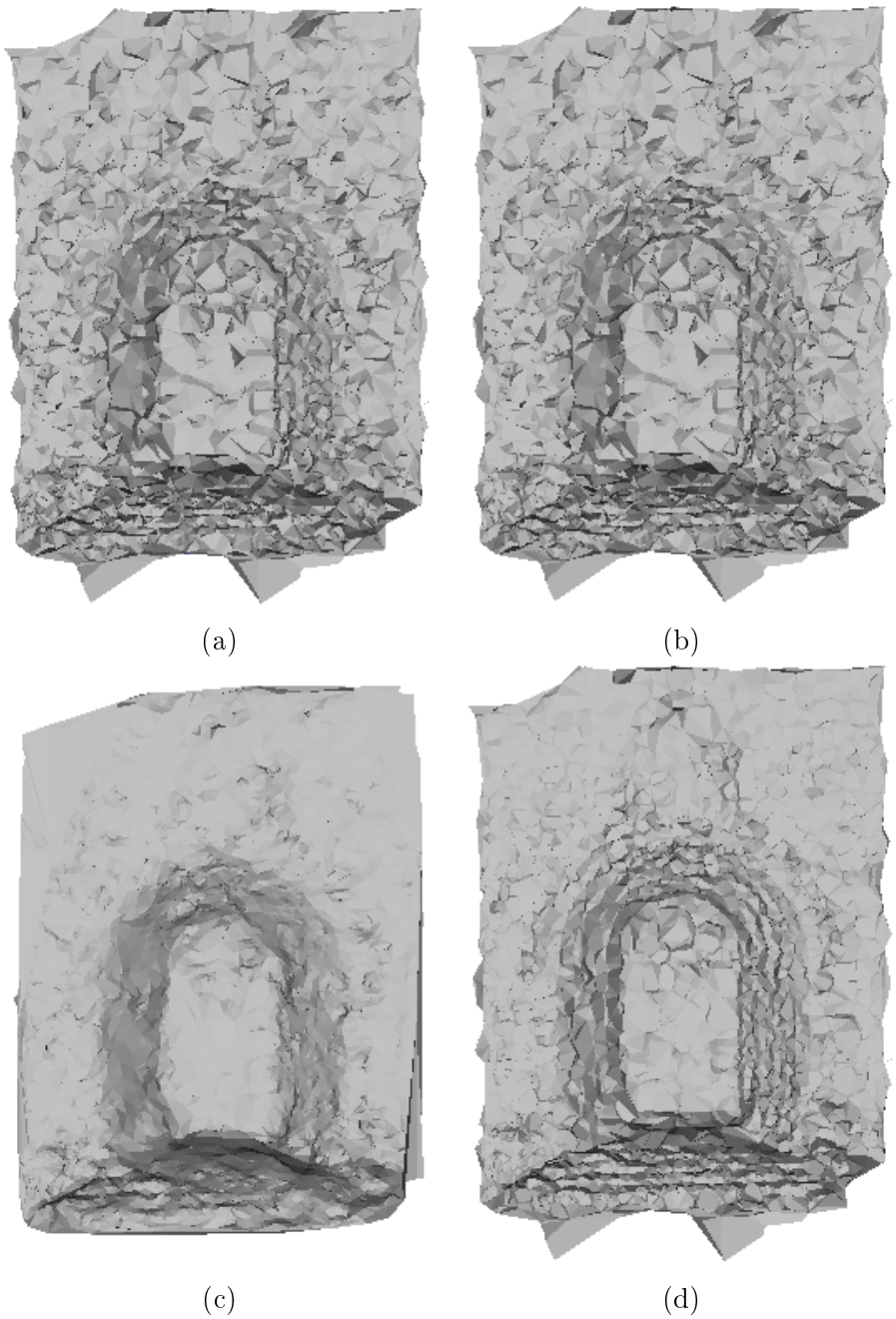


Figure 5.4: Different meshes obtained by: a) The multi-view reconstruction method [Labatut et al., 2007] b) The reconstruction algorithm [Mederos et al., 2005] c) Laplacian Smooth d) our method.

Figure 5.5: Mesh obtained by the method of [Pons et al., 2007] initialized by the output of: a) our method b) The multi-view reconstruction method [Labatut et al., 2007]. c) Some images of the Herz-Jesu-P8 dataset. We have used only a partial cut of the original images.

5.6 Future Works

5.6.1 Dynamic Photo-consistent Surface Reconstruction from Spatio-temporal Noisy Point Clouds

Following our works on the problem of dynamic reconstruction presented in chapters 3 and 4, we can extend the proposed photo-consistent surface reconstruction method to non-rigid dynamic scenes. In that case, a sequence of point clouds, typically featuring high level of noise and high proportions of outliers, are extracted from multiple video sequence providing a four-dimensional spatio-temporal cloud in $\mathbb{R}^4$. The 4D space is indeed obtained by regarding time as the fourth dimension, and considering a scaling factor v, homogeneous to a speed, between space and time in order to keep physical homogeneity. The problem can be simply extended to 4D, considering the same definitions of medial axis transform, poles and power crust.

Let $\mathcal{M} = \{M_1, \ldots, M_n\} \in \mathbb{R}^4$ be the set of n four-dimensional sampled points on all time instants, and $\mathcal{B} = \{B_1, \cdots, B_m\}$ the set of m polar balls of their corresponding 4D Voronoi diagram. Similarly to the static case, we define $\vec{l} = \{l_1, \cdots, l_m\} \in \{-1, 0, +1\}^m$ a labeling vector describing the state of the poles: a pole is labeled -1 if it is at the interior, or $+1$ if it is at the exterior of the spatio-temporal hypersurface. A pole labeled 0 is considered as noise and will be discarded in the estimation of the medial axis transform. $\mathcal{S}_{\mathcal{B}}(\vec{l})$, the inversion of the medial axis transform is a polyhedral hypersurface in $\mathbb{R}^4$, constructed by the faces of the 4D power diagram of $\mathcal{B}$ which lie between inner and outer poles according to $\vec{l}$. In order to estimate the best hypersurface representing the dynamic scene, we integrate photo-consistency/smoothness measure over all faces of $\mathcal{S}$. The photo-consistency measure, however, should be computed in each frame and summed over all the sequence since it is only available in discrete time instants. The global spatio-temporal energy follows,

$$\mathrm{E}(\mathcal{S}_{\mathcal{B}}(\vec{l})) = \sum_{\bar{F} \in \mathcal{S}_{\mathcal{B}}(\vec{l})} \mathrm{area}(\bar{F}) + \sum_{\left(F \in (\mathcal{S}_{\mathcal{B}}(\vec{l}) \cap H), H\right)} \gamma(F) \tag{5.6}$$

where H is a temporal plane, $\bar{F}$ is a 4D facet of the hypersurface $\mathcal{S}$ in $\mathbb{R}^4$, and F is a 3D facet of the surface $(\mathcal{S}_{\mathcal{B}}(\vec{l}) \cap H)$.

Minimizing the first term, which is the area of the 4D hypersurface $\mathcal{S}$, encourages the spatio-temporal smoothness of the hypersurface. The second term measures how well the three-dimensional slices of the hypersurface $\mathcal{S}$ match the different input images at their corresponding frame.

Similarly to the static case, the optimal hypersurface is computed by minimizing E. Given a time instant t, the final 3D mesh can then be extracted from $\mathcal{S}$ by

intersecting this hypersurface with a temporal plane at t.

The feasibility of the optimization remains however questionable. The major difference between the static and dynamic energies is the computation of the 3D slices of $\mathcal{S}$ in the latter. This can be computationally very expensive when repeated at each step of an iterative optimization algorithm such as simulated annealing. Fortunately, this is not a problem here, since according to the reasoning of the section 5.3.2, which holds in any dimension, only a few pentatopes (4-simplices in $\mathbb{R}^4$) should be intersected by temporal planes when the state of a pole is modified. As described before, this is limited by the number of neighbor cells of the pole. The simulated annealing can therefore be used here. In the future, we plan to apply our method to multi-view dynamic datasets, through an optimized implementation of our algorithm in $\mathbb{R}^4$.

5.6.2 Surface Reconstruction from Noisy Point Clouds using Power Distance

Another way to represent noisy data is by assigning spatial probability density functions on input points. *Probabilistic concept of distance* can then be naturally derived, yielding interesting theoretical and experimental results on the problem of surface reconstruction. In this section, we propose a novel concept of closeness and we show how it can be plugged in reconstruction methods in order to correctly reconstruct the surface even in presence of uncertainty. Our two-dimensional experiments show the ability of our approach to handle noisy dataset. In the future, we plan to employ this method on real 3D point clouds obtained by laser scan or multiview triangulation.

5.6.2.1 Probabilistic Concept of Closeness

Voronoi diagram of noisy point sets have been introduced in [Aurenhammer et al., 1991], generalizing the definition of the Voronoi diagram to the situation when the position of each point is uncertain and it is described by some density function in space. Let $\mathcal{M} = \{M_1, \ldots, M_n\} \in \mathbb{R}^d$ be a set of n points, each of them distributed by a probability density function ρ_i. Two cases are distinguished according to the probabilistic concept of closeness:

1. Point $X \in \mathbb{R}^d$ is *closer* to M_i than to M_j if the probability of M_i being closer to X is bigger than the probability of M_j being closer to it.

$$\begin{gathered} d_{\text{prob}}(X, M_i) < d_{\text{prob}}(X, M_j) \\ \Leftrightarrow \\ \int\limits_{\mathbb{R}^d} \rho_j(\vec{x}) \int\limits_{B(X,d(X,\vec{x}))} \rho_i(\vec{y}) \mathbf{d}v_y \mathbf{d}v_x > \int\limits_{\mathbb{R}^d} \rho_i(\vec{x}) \int\limits_{B(X,d(X,\vec{x}))} \rho_j(\vec{y}) \mathbf{d}v_y \mathbf{d}v_x \end{gathered} \tag{5.7}$$

   where $B(C, r)$ is the ball centered at C with radius r. Using this concept of closeness, X belongs to the Voronoi cell of M_i if the probability of M_i being closest to it is maximal.

2. Point $X \in \mathbb{R}^d$ is *closer* to M_i than to M_j if the expected distance of M_i to X is smaller than the expected distance of M_j to it.

$$\begin{gathered} d_{\text{prob}}(X, M_i) < d_{\text{prob}}(X, M_j) \\ \Leftrightarrow \\ \int\limits_{\mathbb{R}^d} d(X, \vec{x}) \rho_i(\vec{x}) \mathbf{d}v_x < \int\limits_{\mathbb{R}^d} d(X, \vec{x}) \rho_j(\vec{x}) \mathbf{d}v_x \end{gathered} \tag{5.8}$$

Using this concept of closeness, X belongs to the Voronoi cell of M_i if the expected distance of M_i from the point X is minimal.

These two probabilistic distances lead to two different types of Voronoi diagram, each of them constructible in $O(n \log n)$ for points with uniform probability density in a circle [Aurenhammer et al., 1991]. Unfortunately, these diagrams are not easily computed, the results are very complicated to be used in real applications. However, as also mentioned in [Aurenhammer et al., 1991], a slightly different concept of expected-closeness yields to a known distance widely studied and employed: the power distance (cf. 5.2.1.1). In this section, we formulate the concept of *noise* and *outlier* by use of the power distance of weighted points, we propose then to employ this distance in surface reconstruction algorithms in order to handle correctly the incertainty in input points.

Uncertainty of a Point Let M be a point in $\mathbb{R}^d$ represented by its probability distribution funcion ρ. For simplification, suppose that the expected position of M is at origin. Consider the expected squared distance of a point $X \in \mathbb{R}^d$ to M,

$$\begin{aligned} \left\langle d^2(X,M) \right\rangle &= \left\langle (\vec{X} - \vec{M})^2 \right\rangle \\ &= \vec{X}.\vec{X} + \left\langle \vec{M}.\vec{M} \right\rangle - 2\vec{X} \cdot \left\langle \vec{M} \right\rangle \\ &= X^2 + \left\langle M^2 \right\rangle \end{aligned} \tag{5.9}$$

Interestingly, this coincides with the squared power distance of the unweighted point X to the weighted point $\hat{M}$ centered at the expected position of M with the power $-\left\langle M^2 \right\rangle$.

In practice, two kinds of uncertainty appear:

- spatial uncertainty
- inlier/outlier uncertainty

The first type is usually modeled by a Gaussian probability distribution centered at the point. The expected squared distance of X to M is therefore,

$$\left\langle d^2(\vec{X}, \vec{M}) \right\rangle = X^2 + \sigma^2$$

where σ^2 is the variance of the Gaussian which corresponds to the probability distribution of M. This is the squared power distance of X and the weighted point $\hat{M} = (M, -\sigma^2)$.

The second type of uncertainty on a point M is modeled by the probability p_{in} of M being inlier. Unfortunately, this is not a density function and cannot be employed

in equation 5.9 directly. However, we propose to interpret an outlier as an infinit point: M is located at origin with probability p_{in}, or at infinity with probability $1 - p_{\text{in}}$. In practice, however, a relatively big constant R shall be used for infinity. The squared expected distance of M to origin is therefore

$$\langle M^2 \rangle = p_{\text{in}}.0 + (1 - p_{\text{in}})R^2 = (1 - p_{\text{in}})R^2 \tag{5.10}$$

The squared expected distance of X to M is therefore the squared power distance of X and the weighted point $\hat{M} = (M, -(1 - p_{\text{in}})R^2)$. Combining the two types of uncertainty, for a point M with spatial uncertainty σ and inlier probability p_{in}, we obtain

$$\langle M^2 \rangle = p_{\text{in}}\sigma^2 + (1 - p_{\text{in}})R^2 \tag{5.11}$$

The expected squared distance of X with M is therefore

$$\langle d^2(X, M) \rangle = d_{pow}(X, \hat{M}) \tag{5.12}$$

where

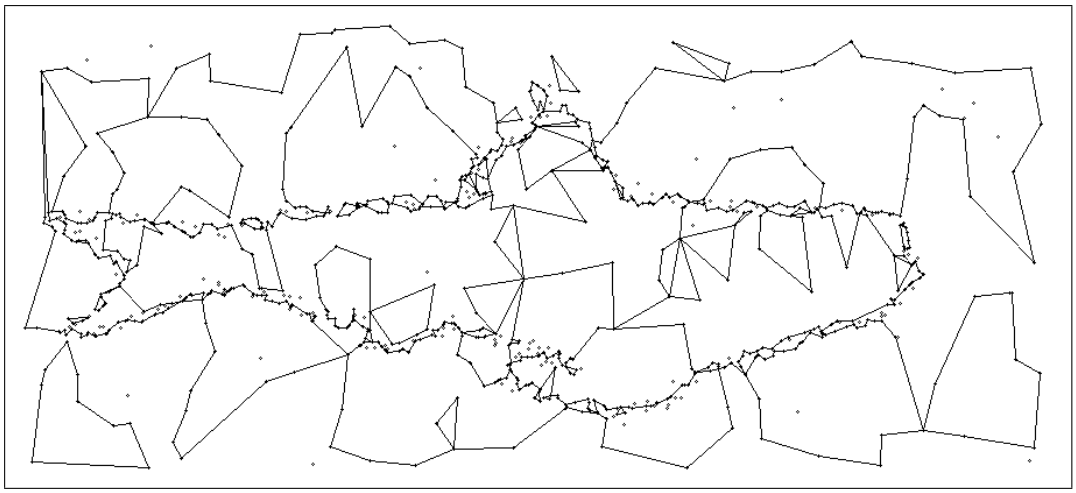
$$\hat{M} = (M, -(p_{\text{in}}\sigma^2 + (1 - p_{\text{in}})R^2)). \tag{5.13}$$

Finally, we suggest that the aforementioned probabilistic distance can be plugged in Delaunay based reconstruction methods, replacing the Delaunay triangulation by a regular triangulation of input points weighted according to their noise level. Indeed, a consequence of the distance 5.12 is that a point can disappear from the regular triangulation if it is not precise enough respecting its neighborhood. In other words, the result depends on the local variation of the noise and not on its value [1].

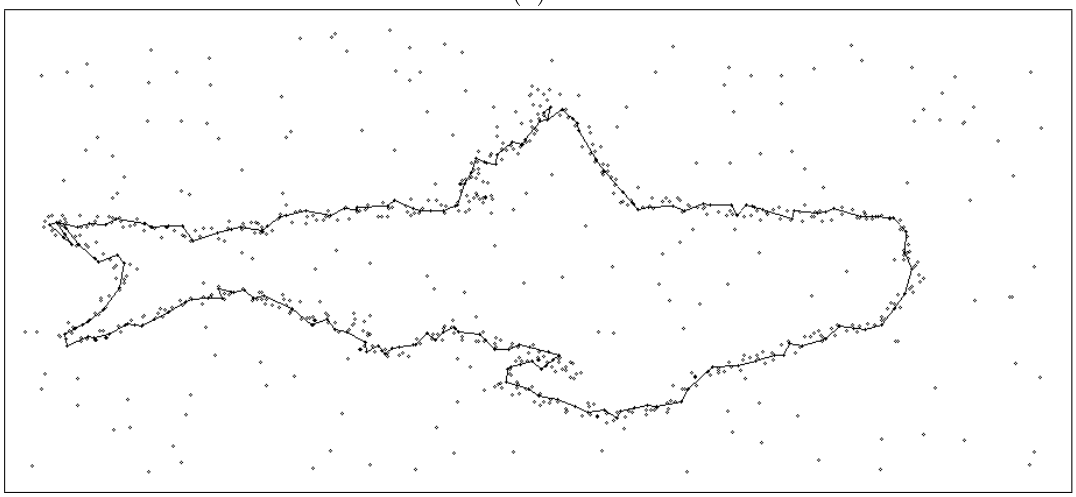
Experimental Results We have tested our method on the *power crust* reconstruction algorithm [Amenta et al., 2001] in $\mathbb{R}^2$. To this end, we have replaced the first Delaunay triangulation computed in the power crust algorithm by a regular triangulation of noisy input points weighted by the equation 5.13. For our experiment, we have added noise to a set of 685 points initially distributed on a curve. 200 outliers, with inlier probability in the range of $[0, 0.6]$ have also been added to the point set. Noise variance of each point has been approximated by a function of point density at its position. Figure 5.6 shows a comparison between the results

[1] Imagine the case where all points have the same incertaitly. The regular triangulation will coincide with the Delaunay triangulation, and all points will appear in the result whatever their noise level. This is intuitively correct since all noisy points have the same level of information and should appear in the result, unless there are some more precise points to be preferred to them.

of the original algorithm (a) and the power crust on the weighted points (b). The results show that despite the spatial uncertainty of the points and the presence of outliers, the power crust approximates correctly the curve when initial points are replaced by weighted points computed by our method. In the future, we plan to test this method on real 3D datasets.



(a)



(b)

Figure 5.6: (a) The power crust of a set of 685 noisy points sampled from a curve, and 200 points uniformly distributed on the plane (outliers). (b) The power crust of the same set of points using our proposed distance.

5.6.3 The Part-time Post Office Problem

The concept of closeness proposed in 5.6.2 does not fully describe the probabilistic distance of noisy points. Particularly, regarding an outlier as an infinite point is questionable. In this section, we define a more realistic concept of closeness, yielding to a complex Voronoi diagram. In the future, we plan to study more of its properties, and we wish to find an algorithm to compute it efficiently.

5.6.3.1 Probabilistic Voronoi diagram

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of n points in $\mathbb{R}^d$, and $\mathcal{P} = \{p_1, \ldots, p_n\}$ their respective presence probability: M_i is present in space (inlier) with probability p_i, or absent (outlier) with probability $1 - p_i$. We define the Voronoi cell of point M_i the set of all points in space whose probability to be closer to M_i than to other points is maximal:

$$V(M_i) = \left\{ X \in \mathbb{R}^d : \forall j, \Pr\left(\forall k, d(X, M_i) \leq d(X, M_k)\right) \geq \Pr\left(\forall k, d(X, M_j) \leq d(X, M_k)\right) \right\} \tag{5.14}$$

Let X be a point in $\mathbb{R}^d$. For X to be closer to M_i than to other points of $\mathcal{M}$, two conditions should be satisfied:

1. M_i should be present
2. all points M_k, for which X is closer to M_k than to M_i, should be absent. These points lie in the interior of the sphere centered at X passing through M_i

The first condition is satisfied with probability p_i. The second condition is satisfied with probability:

$$\prod_{M_k \in B(X, d(X, M_i))} (1 - p_k)$$

where $B(C, r)$ is the ball centered at C with radius r. As a result, X belongs to the Voronoi cell of M_i iff

$$\forall j,\ p_i \prod_{M_k \in B(X, d(X, M_i))} (1 - p_k) > p_j \prod_{M_k \in B(X, d(X, M_j))} (1 - p_k) \tag{5.15}$$

The probabilistic distance of a point X to M_i can therefore be defined by:

$$d_{\text{prob}}(X, M_i) = -p_i \prod_{M_k \in B(X, d(X, M_i))} (1 - p_k) \tag{5.16}$$

Note that the probabilistic distance between X and M_i depends also on the probability and the position of the other points M_k. Now we prove that a cell of Voronoi diagram constructed by this distance, is a cell complex, which is a subset of the *arrangement* of the pairwise bisector hyperplanes of all points. The arrangement of a set of hyperplanes $\mathcal{H}$ is defined as follows,

Definition. *Let $\mathcal{H}$ be a set of n hyperplanes in $\mathbb{R}^d$. $\mathcal{H}$ induces a decomposition of space into a collection of bounded or unbounded polytopes with pairwise disjoint interiors. These polytopes and their faces form a cell complex which is called the arrangement of $\mathcal{H}$.*

Theorem 5.6.1. *Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of n points in $\mathbb{R}^d$, and $\mathcal{P} = \{p_1, \ldots, p_n\}$ their respective presence probability. Let F a face of the arrangement of the pairwise bisector hyperplanes of $\mathcal{M}$. Then F is a subset of a probabilistic Voronoi face of $\mathcal{M}$: $\exists M_i : F \subset V(M_i)$*

Proof. Let $M_i \in \mathcal{M}$, and X_1 and X_2 two points in F. We prove that $d_{\text{prob}}(X_1, M_i) = d_{\text{prob}}(X_2, M_i)$.

Let M_j another point in $\mathcal{M}$. F is the intersection of a finite number of half-spaces bounded by bisectors of the the points in $\mathcal{M}$. Consider the half-space $\tilde{H}$ bounded by the bisector of M_i and M_j containing F. We have $X_1 \in \tilde{H}$ and $X_2 \in \tilde{H}$ since they belong both to F. This implies that

$$M_j \in B(X_1, d(X_1, M_i)) \Leftrightarrow M_j \in B(X_2, d(X_2, M_i))$$

So

$$\begin{aligned} d_{\text{prob}}(X_1, M_i) &= -p_i \prod_{M_j \in B(X_1, d(X_1, M_i))} (1 - p_j) \\ &= -p_i \prod_{M_j \in B(X_2, d(X_2, M_i))} (1 - p_j) \quad = \quad d_{\text{prob}}(X_2, M_i) \end{aligned}$$

All points in F belong therefore to the same Voronoi cell. □

The complexity of the probabilistic Voronoi diagram of a set of points is at most the complexity of the arrangement of the pairwise bisectors of the points. This is given by the following theorem.

Theorem 5.6.2. *Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of n points in $\mathbb{R}^d$, and $\mathcal{P} = \{p_1, \ldots, p_n\}$ their respective presence probability. The complexity (namely, the number of faces) of the probabilistic Voronoi diagram of $\mathcal{M}$ with respect to the distance defined by 5.16 is $O\left(n^{d+2}\right)$.*

Proof. n points make $O(n^2)$ bisectors. The result follows as a consequence of theorem 5.6.1, since the total number of faces in a simple arrangement of n hyperplanes is $O(n^d)$. □

Example Figure 5.7 shows an example of the probabilistic Voronoi diagram of 10 points in plan. Points are shown by circles with radius proportional to their existance probability, and the Voronoi cell of each point is colored by the color of the point. Note that Voronoi cells are not necessarily convex, and the points do not necessarily belong to their corresponding cells.

5.7 Publication

This work (apart from section 5.6 which is in progress) has been published in ICIP conferance [Aganj et al., 2009a],

- E. Aganj, R. Keriven and J.-P. Pons. *Photo-consistent surface reconstruction from noisy point clouds.* In IEEE International Conference on Image Processing, 2009.

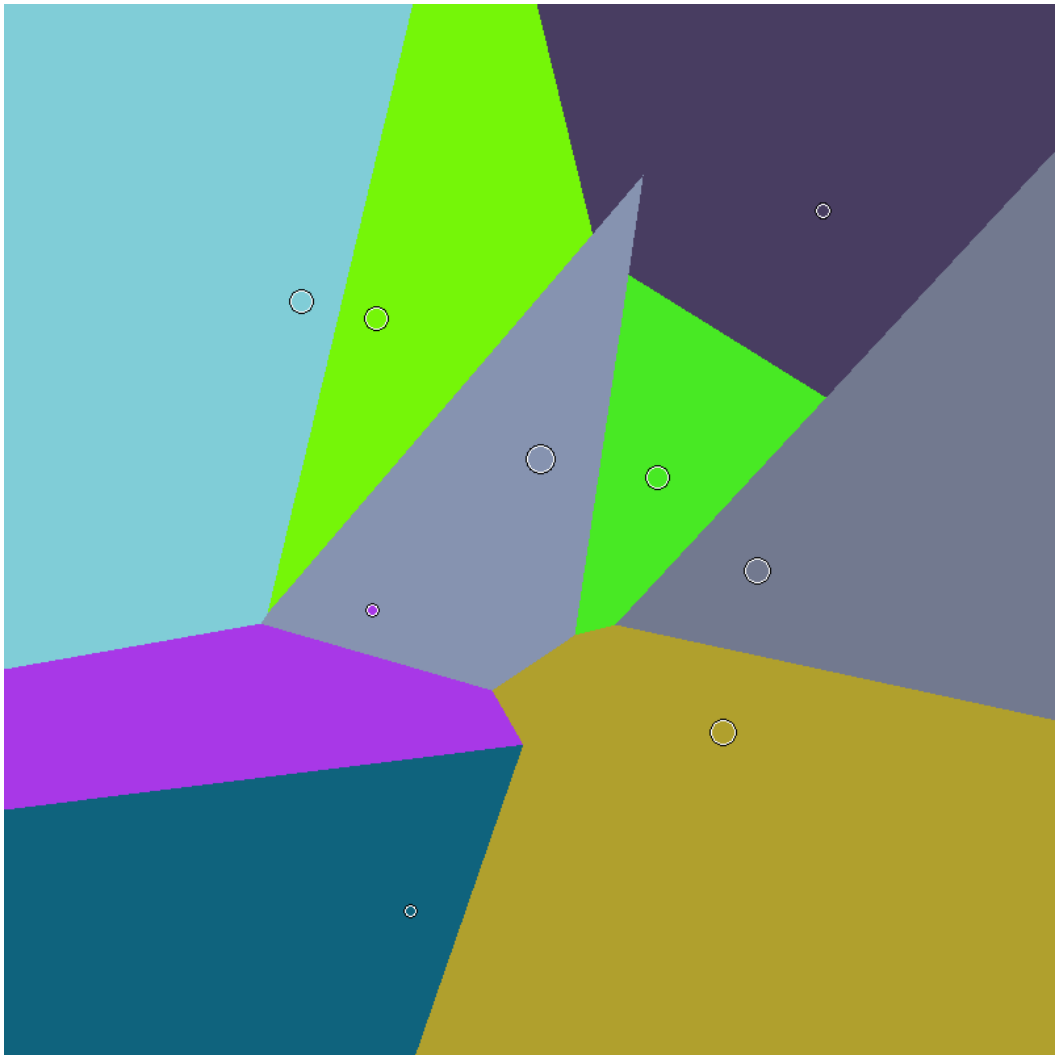


Figure 5.7: The probabilistic Voronoi diagram of 10 points in plan. Points are shown by circles with radius proportional to their existence probability. The Voronoi cell of each point is colored by the color of the point. Note that Voronoi cells are not necessarily convex, and the points do not necessarily belong to their corresponding cells.

Part II

Multi-view Texturing

Chapter 6

Multiview Texturing

Abstract

Reprojection of texture issued from cameras on a mesh estimated from multi-view reconstruction is often the last stage of the pipeline, used for rendering, visualization, or simulation of new views. Errors or imprecisions in the recovered 3D geometry are particularly noticeable at this stage. Nevertheless, it is sometimes desirable to get a visually correct rendering in spite of the inaccuracy in the mesh, when correction of this mesh is not an option, for example if the origin of error in the stereo pipeline is unknown, or if the mesh is a visual hull. We propose to apply slight deformations to the data images to fit at best the fixed mesh. This is done by intersecting rays issued from corresponding interest points in different views, projecting the resulting 3D points on the mesh and reprojecting these points on the images. This provides a displacement vector at matched interest points in the images, from which an approximating full distortion vector field can be estimated by thin-plate splines. Using the distorted images as input in texturing algorithms can result in noticeably better rendering, as demonstrated here in several experiments.

Contents

6.1 Introduction

Recovering 3D geometry from multi-view images or videos is the focus of the stereo research community in computer vision, robotics, and photogrammetry [Atkinson, 2001]. Usage dictates the requirements and priorities about accuracy of the estimated depth information: from rough precision for obstacle avoidance in robot navigation to highly precise and controlled measurements in telemetry and surveying. Despite years of research and development of computing capacities, and whereas the mathematical foundations are well understood [Faugeras and Luong, 2001, Hartley and Zisserman, 2003, Ma et al., 2004], the required precision is not always practically reachable, which may be due to faulty calibration (uncorrected geometric distortion, imprecise focal position), approximations (interpolation of disparity in non-textured regions), or plain errors of algorithms in presence of unexpected conditions (specular surfaces, transparency, etc). Also several stereo pipelines include a step of global, non-convex energy minimization, as for example [Keriven and Faugeras, 1998, Pons et al., 2007, Vu et al., 2009]. As they typically involve a gradient descent scheme, they are susceptible of stopping at a local minimum and have no way of recovering a better 3D geometry. Other methods involve a careful succession of heuristics to refine a visual hull obtained from silhouettes, as for example [Furukawa and Ponce, 2007a]. The base hypotheses of such heuristics may also be somewhat in default. In other cases, the visual hull is used directly for efficiency reasons. Whereas the resulting information may be unusable for precise scientific measures, it may still be useful and sufficient in motion capture for example. In that case, the rendering should mask as best as possible the incorrect geometry.

While algorithms exist that select the image to use as texture on each part of the mesh to minimize illumination change artifacts, they assume that the images are compatible with the mesh. In our case, that assumption does not stand and we must do correct rendering in spite of these inconsistencies. As the mesh is already the result of an optimization, it cannot be refined. The only possibility is to modify the images themselves. This is the approach of Eisemann *et al.* in [Eisemann et al., 2008]. The authors warp the input images by aligning reprojected images through optical flow estimation, for which they use a near-real-time GPU implementation. By contrast, we propose to use feature points as tie points for the registration of images, and to warp the images following a thin-plate spline approximation of the displacement field. Computational cost is normally low, as correspondence of tie points is often already computed and used earlier in the stereo pipeline to estimate epipolar geometry.

Recent work of Tzur and Tal [Tzur and Tal, 2009] is an interesting approach to

the problem. The model is assumed to fit imperfectly with the image, and given a set of projected vertices, a local projection matrix is estimated. The final warp is a weighted average of these local maps. Notice however that the method requires manual input of some projected vertices of mesh in the image. An interactive software specialized to plant modeling is also described in [Quan et al., 2009].

The rest of this chapter is organized as follows. Section 2 describes the details of our algorithm and the required mathematical foundation. Section 3 shows experimental results of this method on diverse data. Finally we draw some conclusions in Section 4.

6.2 Morphing images to adapt to the mesh

6.2.1 Overview of the algorithm

Instead of correcting the mesh to fit the input images, which we assume we cannot do as the mesh is already obtained as some optimum, we correct the input images to fit the output mesh. We suppose that the camera positions, orientations and internal parameters, so as the mesh, are all correct, and we look for deformations in each input image to fit them. This is done in 4 steps:

1. Find matching points in different views.
2. Project on the mesh the obtained 3D points and reproject them onto the views.
3. Approximate the resulting sampled vector field in each image and deform them accordingly.
4. Use a multi-view texturing algorithm for rendering.

Notice that the match points detection is often already done as a first step in the stereo pipeline for calibration, therefore this entails no additional computation. Next sections give details on these different steps.

6.2.2 Interest point matching

Detection of points that have a non-ambiguous local neighborhood has seen remarkable progress in the last few years. They are some kind of generalized extrema or corners. Most of these encode their neighborhood with a similarity invariant signature, although affine invariance can be partly accommodated. Most popular of those are SIFT [Lowe, 2004], which correspond to local extrema in the Gaussian pyramid, or generalized corners, and MSER [Matas et al., 2002], which are centroids of contrasted upper or lower level sets of the image radiometry. Any type of feature points

can be used to match between different views [Mikolajczyk et al., 2005]. We use SIFT points in our experiments, although MSER would also fit.

As noticed above, the interest points are already computed for calibration of the stereo system, and provide 3D point clouds for the initial mesh. However the mesh is often subsequently modified by some smoothing procedure, and then the 3D points are not anymore on the mesh. The next step measures this difference to adapt the images.

6.2.3 Reprojection of 3D points through the mesh

Reprojection is illustrated in Fig. 6.1. Intersecting rays passing through matching feature points via the respective focal points yields the 3D point position. Ideally, these rays would intersect in 3D, but because of imprecise calibration or imprecise detection they may not[1]. The least squares error solution is the 3D point that minimizes the sum of square distances to the rays and can be computed by a closed formula. Such a point is expected to be on the mesh, but because of the imprecision of the mesh, it may reside nearby. A natural adjustment is to project the 3D point M on the mesh, yielding a point $\tilde{M}$. We can then assume that $\tilde{M}$ is the real 3D position and that the images are faulty. We reproject $\tilde{M}$ on the images where it has been observed, yielding corrected positions of the feature points.

Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be the set of n 3D points triangulated from at least two feature points. Let C_i be the set of views from which the point M_i has been generated. The deformation vectors of image j are computed as

$$\mathcal{V}_j = \left\{ \left(\Pi_j(M_i), \Pi_j(\tilde{M}_i) \right) | M_i \in \mathcal{M}, j \in C_i \right\} \tag{6.1}$$

where Π_j is the camera projection of the view number j, and $\tilde{M}_i$ is the projection of M_i on the 3D surface. To ignore outliers, we simply reject the 3D points that are too far from the mesh. Otherwise, a single large mishap can distort the applied warping and ruin the correction effect.

6.2.4 Dense deformation

Previous step indicates the desired position of matched feature points so that they correspond to 3D points on the mesh. However we need a dense deformation of each image to accommodate these displacements. In other words, in each image we are looking for an interpolation or approximation of a vector field irregularly sampled. A standard technique for that is using thin-plate splines [Bookstein, 1989, Wahba,

[1] Bundle adjustment would try to enforce these intersections as best as possible.

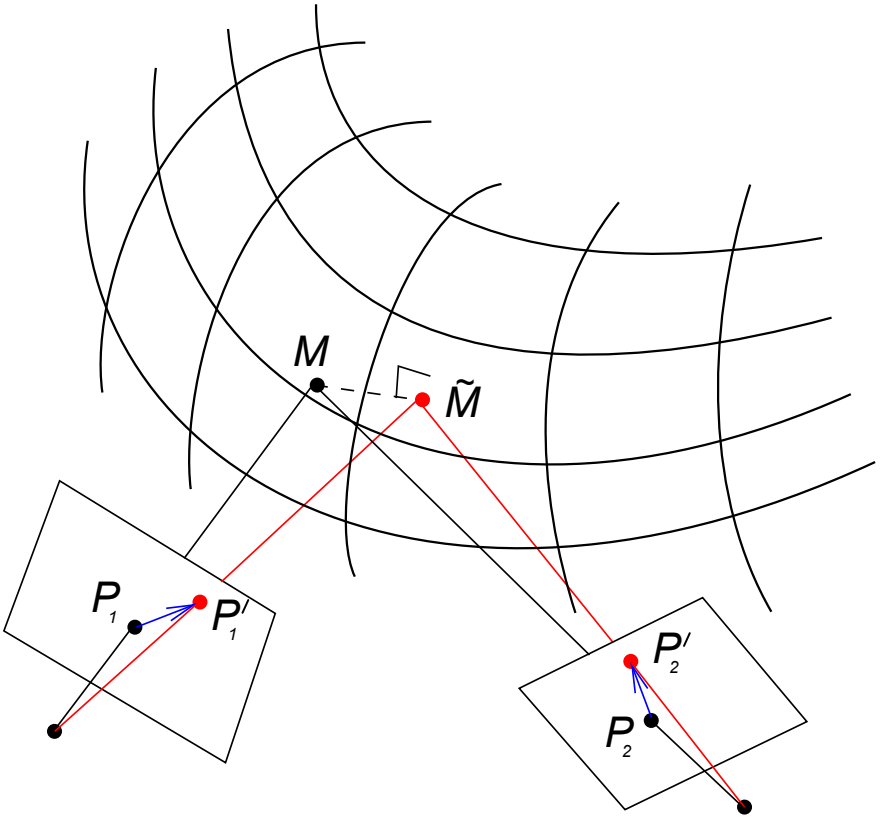


Figure 6.1: Projection of 3D points through the mesh. Corresponding feature points P_1 and P_2 allow to recover a 3D point M. This point is projected on the mesh $\tilde{M}$, which would be observed at P_1' and P_2'.

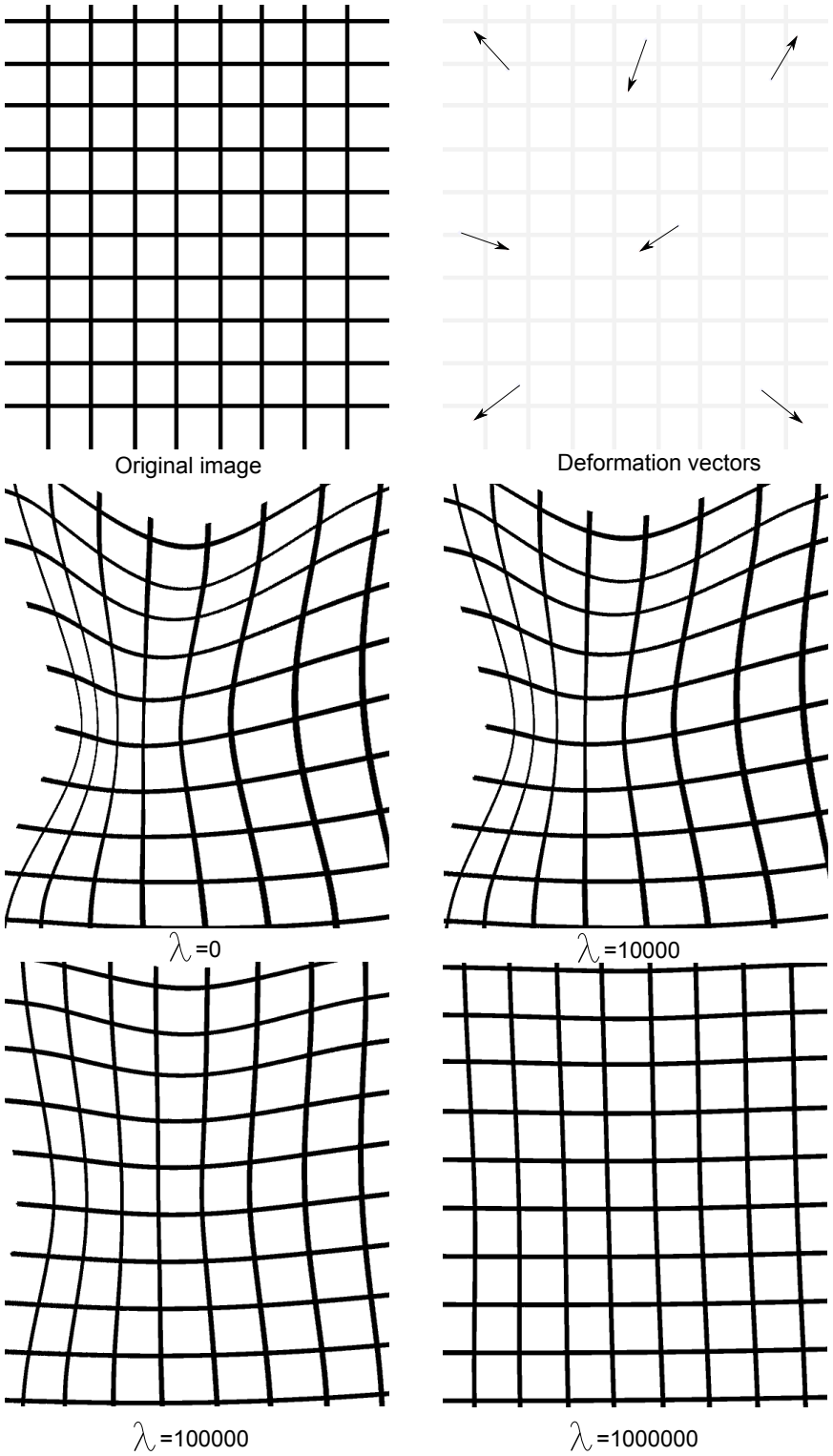


Figure 6.2: An image deformed by the thin-plate spline with different values of λ. The deformation becomes nearly an affine transformation for big λ.

1990]. Given the n feature points P_i and their reprojected positions P'_i through the mesh, thin-plate splines minimize the energy:

$$E(f) = \sum_i \|P'_i - f(P_i)\|^2 + \lambda \int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 dx\, dy \quad (6.2)$$

with a 2-variable function f of the form

$$f(z) = Az + \sum_i \Phi(\|z - P_i\|) w_i \ , \quad (6.3)$$

with A a plane affine transform, Φ a kernel function, usually $\Phi(r) = r^2 \log r$, and w_i a list of 2-vectors representing the non-affine part of the transform. The parameter λ controls the rigidity of the deformation: when $\lambda \to \infty$ the deformation becomes an affine transformation.

Defining K as the $n \times n$ symmetric matrix with entries $K_{ij} = \Phi(\|P_i - P_j\|)$, P as the $3 \times n$ matrix whose column j is composed of homogeneous coordinates of P_j, $P(:, j) = (x_j, y_j, 1)^T$, and P' the $2 \times n$ matrix whose column j is composed of Cartesian coordinates of P'_j, we minimize:

$$E(A, W) = \|P' - AP - WK\|^2 + \lambda \operatorname{trace}(WKW^T) \ ,$$

with A the 2×3 affine transform matrix and W the $2 \times n$ concatenation of the w_j written in columns. The involved norm is the Frobenius norm

$$\|X\|^2 = \operatorname{trace}(X^T X) = \sum_{i,j} X_{ij}^2$$

that is the sum of square coefficients of X, associated to the scalar product $< X, Y >= \operatorname{trace}(X^T Y)$.

Equating to 0 the gradients of E (relative to this scalar product), with respect to A and W, yields:

$$(P' - AP - WK)P^T = 0 \quad (6.4)$$

$$(P' - AP - WK)K + \lambda WK = 0 \quad (6.5)$$

Using the QR decomposition of $P^T = Q_1 R$ (see [Golub and Van Loan, 1996]), let Q_2 be any $n \times (n-3)$ matrix such that $\begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ is orthogonal. Right-multiplying (6.5) by $K^{-1}Q_2$, we get

$$P'Q_2 - WKQ_2 + \lambda WQ_2 = 0 \ ,$$

so that

$$WQ_2 = P'Q_2(Q_2^T K Q_2 - \lambda I)^{-1}$$

and since $WQ_1 = 0$, obtained by substituting $-\lambda W$ to the left factor of (6.4),

$$W \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} = \begin{pmatrix} 0 & P'Q_2(Q_2^T K Q_2 - \lambda I)^{-1} \end{pmatrix} ,$$

which, right-multiplied by $\begin{pmatrix} Q_1 & Q_2 \end{pmatrix}^T$, yields:

$$W = P'Q_2(Q_2^T K Q_2 - \lambda I)^{-1} Q_2^T .$$

Finally, right-multiplying (6.5) by $K^{-1}Q_1$ gives

$$P'Q_1 - AR^T - WKQ_1 = 0 ,$$

whence the solution for A:

$$A = (P' - WK)Q_1 R^{-T} .$$

Figure 6.2 shows an image deformed by the thin-plate spline with different values of λ. The deformation becomes nearly an affine transformation for big λ.

In our experiments, we used an open-source `C++` implementation of thin-plate spline, available at `http://elonen.iki.fi/code/tpsdemo/`.

6.2.5 Texture mapping

Mapping textures from several views on the mesh can be achieved by several methods. Projecting all images on the mesh and doing some weighted averaging, as for example in [Bernardini et al., 2001], leaves some artifacts, such as ghosting. Other methods extract an atlas of the mesh, where each region of the mesh gets its texture from one unique view. The challenge is then to reduce seams visibility. The atlas can be computed by formulating the problem as a Markov Random Field energy minimization [Lempistky and Ivanov, 2007] and then masking the contrast changes between the view by multiband blending at the seams, generalizing work of Burt and Adelson [Burt and Adelson, 1983]. This is the strategy presented in [Allène et al., 2008], which we use in our experiments.

6.3 Experiments

We first demonstrate the proposed algorithm using simulated wrong 3D geometry. The image data are courtesy of R. White *et al.*, who used them in [White et al., 2007]. The 3D geometry was estimated by Furukawa and Ponce [Furukawa and

Ponce, 2007b]. To this 3D model, we apply artificially a translation in space before texturing by the algorithm of [Allène et al., 2008]. For each point on the mesh, the texture comes from one single view (the most frontal one), so that errors can only be seen at transitions from one view to another. Still the benefits of our mesh reprojection algorithm are visible in Fig. 6.3. The warping effect is most visible in the sides of the image.

In the next experiment, we use image data courtesy of J. Starck[2]. The 3D geometry was estimated by visual hull from silhouettes (using an implementation of the algorithm of Franco and Boyer [Franco and Boyer, 2003]) and refined using Poisson surface reconstruction [Kazhdan et al., 2006]. Texturing is done with the algorithm of [Allène et al., 2008] slightly modified to enhance errors: instead of selecting one pixel value, issued from the "best" view, to any point on the mesh, the average of the *two* best views is used. Only the 3 front views were used in the texturing process. This produces blur at misregistered points, otherwise the errors can only be observed at transitions between different cameras in the atlas, which is still noticeable but less striking. Notice that the original images produce artifacts on the arms and on the dancer's left hand, while the dress exhibits some wrong texture. Most of these problems are fixed by the warping, except on some part of the left arm.

6.4 Conclusion

When the multi-view reconstruction pipeline yields an imprecise mesh, we have shown how the input images themselves can be modified to mask the imprecisions in the rendering. Mapping these images as texture on the mesh limits the visible artifacts. Reversing the problem by changing the input (the images) to match the erroneous output (the mesh) does not allow better measurements, but if only a visually pleasing rendering is sufficient, as for example in motion capture for computer generated imagery, this technique provides a simple solution. The algorithm was demonstrated on simulated and real imprecise meshes. Extension of this work to dynamic multi-view stereo (3D+time) using similar algorithms will be investigated.

6.4.1 Future work

Our method can be adapted to spatio-temporal texturing of dynamic scene, employing the extension of thin-plate spline deformation to 3D. Indeed, a frame-by-frame texturing of an imprecise dynamic model by the proposed static method does not

[2] `http://personal.ee.surrey.ac.uk/Personal/J.Starck/`

exploit temporal coherence, leaving still virtual artifacts such as temporal discontinuity of texture. Following our works on the problem of dynamic reconstruction presented in chapters 3 and 4, our method is applicable in the 3D image space obtained by regarding time as the third dimension, and considering a scaling factor v, homogeneous to a speed, between space and time to keep physical homogeneity.

By adding time to the image space, we consider each input video, captured from a viewpoint, as a 3D image whose values are known at discrete time instants. The 3D feature points are naturally constructed by adding time as the third dimension, and the 3D point clouds are computed, as in the static case, by triangulation. The rest of our method is simply extended to 3D: The deformation sample vectors are obtained by projecting the triangulated points on the surface, and then reprojecting them on the images. The deformation vector fields are computed by the 3D thin-plate spline algorithm, using the sampled vectors. The 2D images are then obtained by employing these deformations on the 3D input images, interpolating values linearly between neighbor images. This is formulated as follows.

Let $\mathcal{M}^k = \left\{M_1^k, \cdots, M_{n_k}^k\right\}$ be the set of n_k points triangulated from the feature points at frame k. Let C_i^k be the set of views from which the point M_i^k has been generated. The deformation vectors of image j are computed as

$$\mathcal{V}_j^k = \left\{ \left(\Pi_j(M_i^k), \Pi_j(\tilde{M}_i^k)\right) \middle| M_i^k \in \mathcal{M}^k, j \in C_i^k, 0 < k \leq K \right\} \tag{6.6}$$

where K is the number of frames, Π_j is the camera projection of the view number j, and $\tilde{M}_i^k$ is the projection of M_i^k on the 3D surface. All $\mathcal{V}_j^k$ can then be used to estimate a 3D deformation vector field for the viewpoint j. In the future, we plan to implement and employ this method on real dynamic datasets to demonstrate its advantages compared to frame-by-frame texturing.

6.5 Publication

This work has been published in ACCV conferance [Aganj et al., 2009b],

- E. Aganj, P. Monasse and R. Keriven. *Multi-View Texturing of Imprecise Mesh*. In Asian Conference on Computer Vision, 2009.

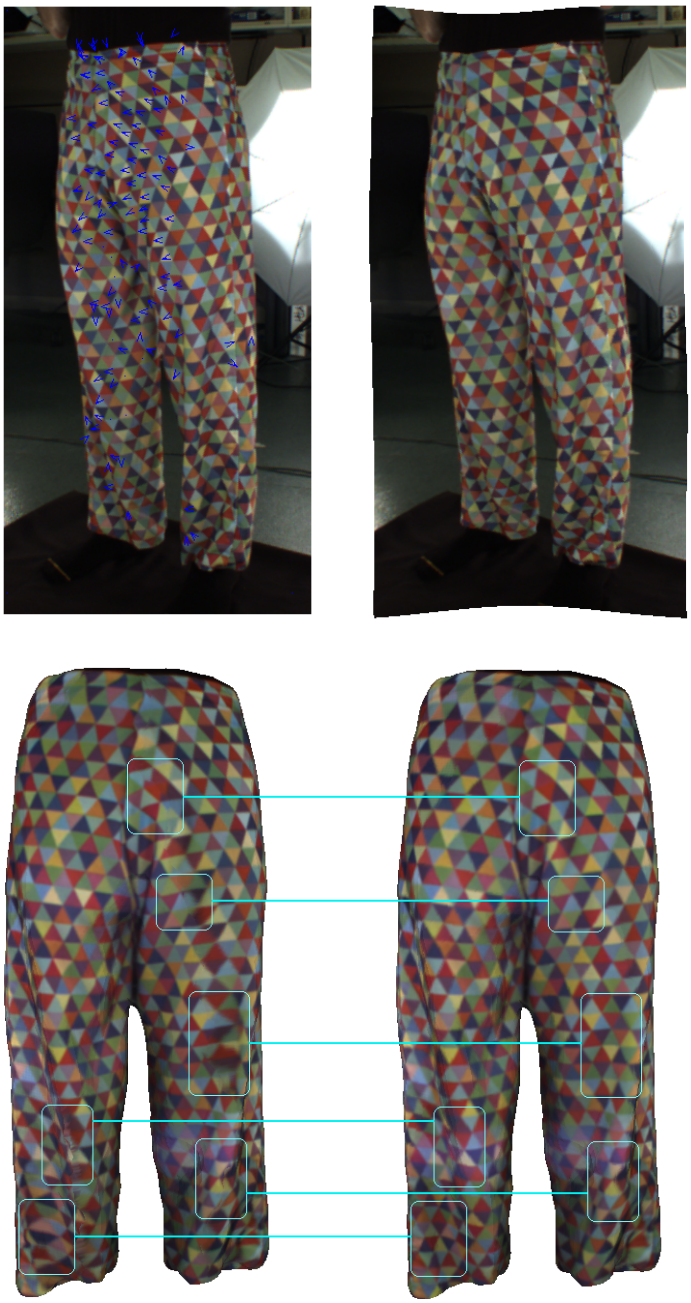


Figure 6.3: Texturing on simulated imprecise mesh ([Furukawa and Ponce, 2007b] plus erroneous deformation). Top: Warping of image to adapt to the mesh. Top left: one original image with displacement vector of key points superimposed. Top right: the warped image using thin-plate spline approximation of this sampled vector field. Bottom: Multi-view texturing on imprecise mesh using [Allène et al., 2008]. Bottom left: texturing with original images. Bottom right: texturing with warped images.

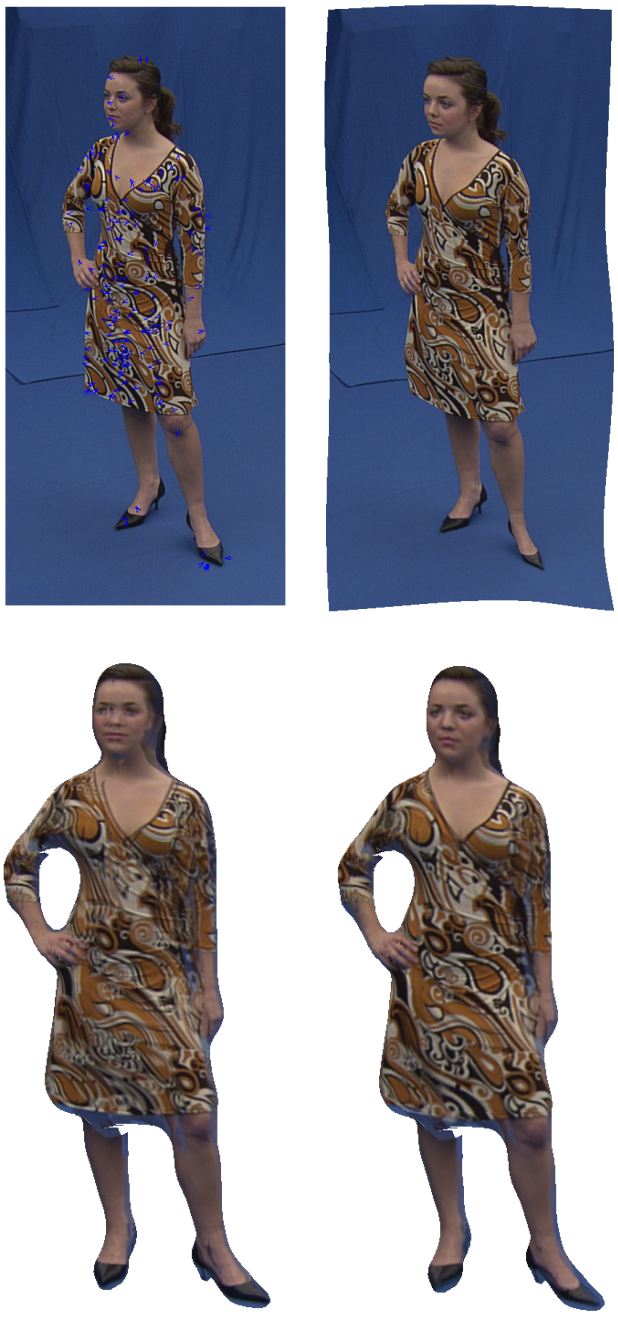


Figure 6.4: Texturing on real imprecise mesh (visual hull from [Franco and Boyer, 2003]). Top: Warping of image to adapt to the mesh. Top left: one original image with displacement vector of key points superimposed. Top right: the warped image using thin-plate spline approximation of this sampled vector field. Bottom: Multi-view texturing on imprecise mesh using [Allène et al., 2008]. Bottom left: texturing with original images. Bottom right: texturing with warped images.

Conclusion

In this thesis, we have considered the problem of multi-view reconstruction, focusing on applicability and flexibility of approaches. In the first part, we have proposed three reconstruction methods sharing the objective of estimating a representation of a static/dynamic scene from a set of multiple images/videos.

- Our first contribution is a method to reconstruct a four-dimensional representation of the spatio-temporal visual hull of a non-rigid scene, outputting a compact and high-quality surface mesh representation of the visual hull, and offering easy and extensive control over the size and quality of the mesh as well as over its associated reprojection error. We have shown that our method has several significant advantages compared to independent frame-by-frame computation of visual hull. It yields a compact representation by exploiting time redundancy, and it offers the possibility of interpolating the shape between consecutive frames thanks to the temporally continuous representation constructed in the four-dimensional space.

- Our second contribution is a multi-view reconstruction method computing a globally optimal spatio-temporal representation of a non-rigid dynamic scene under uncontrolled imaging conditions: it is robust to noise, it does not require any initialization, it does not require silhouettes, it handles closed as well as open scenes with cluttered backgrounds, it exploits visibility information to guide the position of the surface, and it computes a compact, and temporally continuous four-dimensional representation, offering several advantages mentioned above compared to independent frame-by-frame reconstruction of the scene.

Both of aforementioned methods reconstruct a four-dimensional surface as a set of tetrahedral facets extracted from the Delaunay triangulation of a 4D point cloud, exploiting consequently the temporal coherence by constructing a compact, temporally continuous and global representation of the dynamic scene, and handling naturally topology changes thanks to the four-dimensional framework. Finally, we believe that the four-dimensional hypersurface representation employed in these

approaches is likely to inspire progress in other applications, such as segmentation of spatio-temporal MRI sequences of the heart in medical imaging.

- Our third contribution is a photo-consistent surface reconstruction algorithm based on the medial axis transform of the scene, incorporating input images in the reconstruction process for a better accuracy and robustness. In this work, we consider the special case of point clouds extracted from calibrated multi-view image datasets, typically featuring high levels of noise and high proportions of outliers. Our method is based on the estimation of the medial axis transform with a global photo-consistency optimization by simulated annealing.

Despite our static formulation of the problem, we have shown that it can be easily extended to a four-dimensional space, efficiently reconstructing a spatio-temporal surface from a set of noisy point clouds extraceted from a dynamic non-rigid scene. Please refer to section 5.6.1 for more details. In the second part of this thesis, we have considered the problem of multi-view texturing, focusing on the visual correctness of the rendering, in spite of high level of inaccuracy in the mesh.

- Our fourth contribution is a method to compute a visually correct texturing of an imprecise mesh. To this end, we have proposed to slightly deform the data images to fit at best the fixed mesh. We employ then the distorted images as input in texturing algorithms to obtain better rendering.

We have also shown that this method can be extended to the case of dynamic scenes, going beyond independent frame-by-frame texturing of meshes (cf. section 6.4.1). In future, we plan to implement and employ this method on real dynamic datasets to demonstrate its advantages compared to frame-by-frame texturing.

All through this thesis, we have validated our methods by several numerical experiments, all of them implemented in C++, and most of them using the computational geometry algorithms library CGAL [Boissonnat et al., 2000b]. CGAL defines all the geometric primitives needed in this thesis and provides excellent algorithms to compute the Delaunay and regular triangulations. Please note that, our numerical results presented in chapter 3 are generated by a generic unoptimized Delaunay triangulation algorithm which works in any number of dimensions (cf. section 3.3.3). In the future, we plan to use a recently proposed new implementation of the incremental algorithm for constructing Delaunay triangulations in any dimension [Boissonnat et al., 2009] to reduce the computational time of our method.

Finally, please note that, at the end of each chapter we have presented and discussed our different future challenges, particularly in sections 5.6.2 and 5.6.3

where two works in progress concerning novel concepts of probabilistic distance have been proposed yielding different methods to reconstruct a surface from noisy point clouds. In future, we plan to validate these methods by employing them on real 3D datasets.

Bibliography

E. Aganj, J.-P. Pons, F. Ségonne, and R. Keriven. Spatio-temporal shape from silhouette using four-dimensional delaunay meshing. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, Oct 2007. 62, 64

E. Aganj, R. Keriven, and J.-P. Pons. Photo-consistent surface reconstruction from noisy point clouds. In *IEEE International Conference on Image Processing*, 2009a. 106

E. Aganj, P. Monasse, and R. Keriven. Multi-view texturing of imprecise mesh. In *Asian Conference on Computer Vision*, 2009b. 121

E. Aganj, J.-P. Pons, and R. Keriven. Globally optimal spatio-temporal reconstruction from cluttered videos. In *Asian Conference on Computer Vision*, 2009c. 81

N. Ahmed, E. de Aguiar, C. Theobalt, M. Magnor, and H.P. Seidel. Automatic generation of personalized human avatars from multi-view video. In *Proc. Virtual Reality Software and Technology*, pages 257–260, 2005. 33, 64

N. Ahmed, C. Theobalt, C. Rössl, S. Thrun, and H.P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 33, 64

P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. *Shape Analysis ans Structuring. Springer*, 2007. 33

C. Allène, J-P. Pons, and R. Keriven. Seamless image-based texture atlases using multi-band blending. In *19th International Conference on Pattern Recognition*, Tampa, US, Dec 2008. 119, 120, 122, 123

J. Aloimonos. Visual shape computation. *IEEE Proc.*, 76(8):899–916, 1988. 21

N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999. 46, 85, 89

N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19: 127–153, 2001. 85, 89, 90, 94, 101

D. Anguelov, D. Koller, P. Srinivasan, S. Thrun, H.-C. Pang, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. *Proceedings of the Neural Information Processing Systems (NIPS) Conference*, 2004. 33

K.B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision.* Whittles Publishing, 2001. 113

D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Annual Symposium on Computational Geometry*, pages 201–210, 2003. 42

D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration.* Springer-Verlag, Mathematics and Visualization, 2009. 87

F. Aurenhammer, G. Stöckl, and E. Welzl. The post office problem for fuzzy point sets. In *CG '91: Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications*, pages 1–11, London, UK, 1991. Springer-Verlag. ISBN 3-540-54891-2. 99, 100

C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22:469–483, 1996. 76

B.G. Baumgart. *Geometric Modeling for Computer Vision.* PhD thesis, Stanford University, 1974. 20

B.G. Baumgart. A polyhedron representation for computer vision. *AFIPS National Computer Conference*, 1975. 24

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of European Conference on Computer Vision*, pages 404–417, 2006. 34

F. Bernardini, I. Martin, and H. Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Trans. on Visualization and Computer Graphics*, 7(4): 318–332, 2001. 119

P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Comput. Surv.*, 17(1):75–145, 1985. ISSN 0360-0300. 21

H. Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967. 87

J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. pages 223–232, 2000. 85

J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67:405–451, 2005. 37, 44, 45, 48

J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of Lipschitz surfaces. In *Annual Symposium on Computational Geometry*, 2006. 44, 45, 48

J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998. 37, 40, 41, 42

J.-D. Boissonnat, O. Devillers, M. Teillaud, and M. Yvinec. Triangulations in cgal (extended abstract). In *SCG '00*, pages 11–18, NY, USA, 2000a. ACM. ISBN 1-58113-224-7. 92

J.-D. Boissonnat, O. Devillers, M. Teillaud, and M. Yvinec. Triangulations in CGAL. In *Annual Symposium on Computational Geometry*, pages 11–18, 2000b. 57, 76, 126

J.D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *Proceedings of the 25th annual Symposium on Computational geometry (SCG'09)*, Aarhus, Denmark, June 2009. ACM. 126

F.L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 (6):567–585, 1989. 115

E. Boyer and J.-S. Franco. A hybrid approach for computing visual hulls of complex objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 695–701, 2003. 22, 25

E. Boyer, M.O. Berger, and C. Inria. 3d surface reconstruction using occluding contours. *International Journal of Computer Vision*, 22:219–233, 1995. 25

E.M. Bronstein, M.M. Bronstein, Student Member, Student Member, R. Kimmel, and Senior Member. Calculus of non-rigid surfaces for geometry and texture manipulation. *IEEE Trans. Visualization and Comp. Graphics*, 13:902–913, 2007. 33

P.J. Burt and E.H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, 2(4):217–236, 1983. 119

R.L. Carceroni and K.N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *Int. J. Comput. Vision*, 49(2-3):175–214, 2002. ISSN 0920-5691. 31

K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221–247, 2004. 64

K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time: Part I: Theory and algorithms. *The International Journal of Computer Vision*, 62(3): 221–247, 2005a. 27

K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time: Part II: Applications to human modeling and markerless motion tracking. *The International Journal of Computer Vision*, 63(3):225–245, 2005b. 27

C.H. Chien and J.K. Aggarwal. Model construction and shape recognition from occluding contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):372–389, 1989. ISSN 0162-8828. 21

R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *Int. J. Comput. Vision*, 9(2):83–112, 1992. ISSN 0920-5691. 24

J.P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, September 1998. 28

E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007a. 34, 64

E. de Aguiar, C. Theobalt, C. Stoll, and H.P. Seidel. Marker-less 3d feature tracking for mesh-based human motion capture. In Ahmed M. Elgammal, Bodo Rosenhahn, and Reinhard Klette, editors, *Workshop on Human Motion*, volume 4814 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007b. ISBN 978-3-540-75702-3. 34

E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *ACM SIGGRAPH*, 2008. 34, 64

M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. SpringerVerlag, 1997. 37

T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Comput. Geom.*, 35(1-2):124–141, 2006. 85

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating Textures. *Computer Graphics Forum (Proc. Eurographics EG'08)*, 27(2):409–418, 4 2008. 113

C.H. Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Comput. Vis. Image Underst.*, 96(3):367–392, 2004. ISSN 1077-3142. 29

O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7: 336–344, 1999. 29

O. Faugeras and Q.T. Luong. *The Geometry of Multiple Images*. MIT Press, 2001. 113

L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962. 67

J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference*, volume 1, pages 329–338, 2003. 23, 25, 26, 56, 57, 58, 59, 61, 120, 123

J.-S. Franco and E. Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. In *International Conference on Computer Vision*, volume 2, pages 1747–1753, 2005. 24

J.-S. Franco, M. Lapierre, and E. Boyer. Visual shapes of silhouette sets. In *3D Processing, Visualization and Transmission*, 2006. 20

Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007a. 113

Y. Furukawa and J. Ponce. Dense patch models for motion capture from synchronized video streams. Technical report, Willow Technical report 02-07, 2007b. 119, 122

Y. Furukawa and J. Ponce. Dense 3D motion capture from synchronized video streams. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 35, 64

R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006. ISSN 0730-0301. 33

B. Goldlücke and M. Magnor. Space-time isosurface evolution for temporally coherent 3D reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 350–355, 2004. 29, 34, 45, 64, 65

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996. 118

V. Granville, M. Krivánek, and J.P. Rasson. Simulated annealing: A proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994. ISSN 0162-8828. 89

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. 67

L. Guan, S. Sinha, J.-S. Franco, and M. Pollefeys. Visual hull construction in the presence of partial occlusion. In *3D Processing, Visualization and Transmission*, 2006. 27

A. Gueziec and R. Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4), 1995. 54, 76

D. Hähnel, S. Thrun, and W. Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *in Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03*, volume 2003, pages 915–920, 2003. 33

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. 113

M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM Press. ISBN 158113374X. 34

K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH Course Notes*, 2007. 33

D. Huber and M. Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21(1):637–650, July 2003. 33

M. Isenburg, Y. Liu, J. Shewchuk, and J. Snoeyink. Streaming computation of Delaunay triangulations. In *ACM SIGGRAPH*, pages 1049 – 1056, 2006. 60

H. Jin, S. Soatto, and A.J. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *The International Journal of Computer Vision*, 63(3): 175–189, 2005. ISSN 0920-5691. 29

M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, 2006. 120

R. Keriven and O. Faugeras. Complete dense stereovision using level set methods. In *5th European Conference on Computer Vision*, 1998. 113

D. Kirsanov and S.J. Gortler. A discrete global minimization algorithm for continuous variational problems. harvard computer science technical report: Tr-14-04. Technical report, Cambridge, MA, 07/2004 2004. 67

J.J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984. 21, 24

R. Kolluri, J.R. Shewchuk, and J.F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing (SGP'04)*, pages 11–21, 2004. 90

V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004. ISSN 0162-8828. 67

P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, Oct 2007. 64, 65, 73, 77, 78, 79, 92, 93, 95, 96

A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. 20, 64

S. Lazebnik, E. Boyer, and J. Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 156–161, 2001. 25

V.S. Lempistky and D.V. Ivanov. Seamless mosaicing of image-based texture maps. In *Proc. of IEEE International Conference on Computer Vision*, 2007. 119

M. Leventon, E. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 316–322, 2000. 52

B. Lok. Online model reconstruction for interactive virtual environments. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 69–72, New York, NY, USA, 2001. ACM. ISBN 1-58113-292-1. 24

W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–170, 1987. 24

D.G. Lowe. Object recognition from local scale-invariant features. *Computer Vision, IEEE International Conference on*, 2:1150–1157 vol.2, 1999. 33

D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 114

Y. Ma, S. Soatto, J. KoseckÃ¡, and S.S. Sastry. *An Invitation to 3-D Vision*, volume 26 of *Interdisciplinary Applied Mathematics*. Springer, 2004. 113

W.N. Martin and J.K. Aggarwal. Volumetric descriptions of objects from multiple views. 5(2):150–158, March 1983. 22

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. *Proceedings of British Machine Vision Conference*, I:384–393, 2002. 114

T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *Comput. Vis. Image Underst.*, 96(3):393–434, 2004. ISSN 1077-3142. 33

W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, and L. McMillan. Image-based visual hulls. In *ACM SIGGRAPH*, pages 369–374, 2000. 25, 26

W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Eurographics Workshop on Rendering*, pages 115 – 126, 2001. 25

B. Mederos, N. Amenta, L. Velho, and L.-H. de Figueiredo. Surface reconstruction from noisy point clouds. In *Symposium on Geometry Processing (SGP'05)*, page 53, Aire-la-Ville, Switzerland, 2005. Eurographics Association. ISBN 3-905673-24-X. 85, 93, 94, 95

A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 0124906303. 33

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21:1087–1092, 1953. 89, 92

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65:2005, 2005. 115

T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268, 2001. 28

J. Neumann and Y. Aloimonos. Spatio-temporal stereo using multi-resolution subdivision surfaces. *International Journal of Computer Vision*, 47(1–3):181–193, 2002. 31

S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988. 45

N. Paragios, Y. Chen, and O. Faugeras. *Handbook of Mathematical Models in Computer Vision*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387263713. 67

S.I. Park and J.K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, July 2006. ISSN 0730-0301. 33

J.-P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *The International Journal of Computer Vision*, 72(2):179–193, Apr 2007. 32, 93, 96, 113

M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics and Image Processing*, 40(1): 1–29, 1987. 23

L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S.B. Kang. Image-based plant modeling. In *Proceedings of ACM SIGGRAPH*, 2009. 114

A. Roche, G. Malandain, X. Pennec, and N. Ayache. The correlation ratio as a new similarity measure for multimodal image registration. pages 1115–1124. Springer Verlag, 1998. 29

M. Rousson and N. Paragios. Shape priors for level set representations. In *European Conference on Computer Vision*, volume 2, pages 78–92, 2002. 52

S. Rusinkiewicz, B. Brown, and M. Kazhdan. 3d scan matching and registration. *International Conference on Computer Vision short courses*, 2005. 33

S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *International Journal of Computer Vision*, volume 35, pages 1067–1073, 1997. 30

J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1983. ISBN 0126372403. 24

J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1694, 1996. 57

D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 345–352, 2000. 23

O. Sorkine. Laplacian mesh processing. *Eurographics Conference*, 2005. 34

J. Starck and A. Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1387–1394, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02. 33

J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007. 34

C. Strecha, W. von Hansen, L.J. Van Gool, and U. Thoennessen P. Fua. On benchmarking camera calibration and multi-view stereo for high resolution imagery. pages 1–8, 2008. 93

S. Sullivan and J. 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. 25

R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing*, 58(1):23–32, 1993. 23

C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, and H.P. Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):663–674, 2007. 64

Y. Tzur and A. Tal. Photogrammetric texture mapping using casual images. In *Proceedings of ACM SIGGRAPH*, 2009. 113

S. Ullman. The interpretation of visual motion. *MIT Press*, 1979. 28

K. Varanasi, A. Zaharescu, E. Boyer, and R.P. Horaud. Temporal surface tracking using mesh evolution. In *European Conference on Computer Vision*, volume 2, pages 30–43, 2008. 34

S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow, 1999. 29, 30

S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6D. *IEEE Conference on Computer Vision and Pattern Recognition*, 2000. 31

S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002. 30

S. Vedula, S. Baker, and T. Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Transactions on Graphics*, 24(2): 240–261, 2005. 30

B. Vijayakumar, D. Kriegman, and J. Ponce. Structure and motion of curved 3d objects from monocular silhouettes. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 327–334, 1996. 27

P. Viola, W.M. Wells, and III. Alignment by maximization of mutual information. *Computer Vision, IEEE International Conference on*, 0:16, 1995. 29

D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3), 2008. 27, 64

H.H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 113

G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990. 115

Y.F. Wang, M.J. Magee, and J.K. Aggarwal. Matching three-dimensional objects using silhouettes. 6(4):513–518, July 1984. 21

A.M. Waxman and J.H. Duncan. Binocular image flows: steps toward stereo-motion fusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):715–729, 1986. ISSN 0162-8828. 28

R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. In *SIGGRAPH*, 2007. 76, 119

G.-S. J. Young and R. Chellappa. 3-d motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness results. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(8):735–759, 1990. ISSN 0162-8828. 28

A. Zaharescu, E. Boyer, and R. Horaud. Transformesh: A topology-adaptive mesh-based approach to surface evolution. In *Asian Conference on Computer Vision*, pages II: 166–175, 2007. 34

D. Zhang and M. Hebert. Harmonic maps and their applications in surface matching. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2524, 1999. ISSN 1063-6919. 33

Y. Zhang and C. Kambhamettu. Integrated 3D scene flow and structure recovery from multiview image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 674–681, 2000. 31

Y. Zhang and C. Kambhamettu. On 3D scene flow and structure estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 778–785, 2001. 31

Z.Y. Zhang and O.D. Faugeras. *3D Dynamic Scene Analysis: A Stereo Based Approach*. Book, 1992. 28

D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics*, 31(Annual Conference Series):259–268, 1997. 32