Contributions to Spatial and Temporal 3-D Reconstruction from Multiple Cameras

Andrei Zaharescu

INSTITUT POLYTECHNIQUE DE GRENOBLE

N^o attribué par la bibliothèque
| | | | | | | | | |

THÈSE

pour obtenir le grade de
Docteur de l'Institut polytechnique de Grenoble

Spécialité : Imagerie, Vision et Robotique
préparée au laboratoire **GRAVIR**
dans le cadre de l'Ecole Doctorale :
Mathématiques, Sciences et Technologies de l'Information, Informatique

———

présentée et soutenue publiquement par

Andrei Zaharescu

le 21 Novembre 2008

Contributions à la reconstruction spatiale et temporelle à partir de plusieurs caméras

———

Directeur de thèse : Radu Horaud
Co-Directeur de thèse : Edmond Boyer

———

JURY

Président :	**Marie-Paule Cani**
Rapporteurs :	**Renaud Keriven**
	Adrian Hilton
Examinateurs :	**Jean-Philippe Pons**
	Slobodan Ilic
Directeur de thèse :	**Radu Horaud**
Co-Directeur de thèse :	**Edmond Boyer**

Contributions to Spatial and Temporal 3-D Reconstruction from Multiple Cameras

Andrei Zaharescu

Dedicată lui Conu' (Mihai Stan) și lui Teicuțu' (Nicolae Roșca).

Abstract

This thesis addresses the necessary steps required to build a framework for spatial and temporal 3-D reconstruction using multiple camera environments: camera calibration and sparse 3-D reconstruction, dense 3-D reconstruction, sparse and dense mesh matching. Firstly, a probabilistic formulation is developed in conjunction with any affine factorization algorithm (3-D point based reconstruction method based on matrix factorization), able to recover both the extrinsic parameters of multiple cameras and 3-D coordinates of control points, given their projected 2-D point correspondences and the intrinsic camera parameters. The proposed framework is robust to outliers and compares favourably with bundle adjustment, a standard non-linear minimization technique, which requires an initial solution not very far from the optimum. Secondly, a provably correct mesh-based surface evolution approach is proposed. It is able to handle topological changes and self-intersections without imposing any mesh sampling constraints. The exact mesh geometry is preserved throughout, except for the self-intersection areas. Sample applications, including mesh morphing and 3-D reconstruction using variational methods, are presented. Thirdly, a scene-aware camera clustering method is developed, able to break large-scale reconstruction tasks in smaller independent partial reconstructions that are memory tractable. Lastly, a new 3 dimensional descriptor is proposed, defined on uniformly sampled triangular meshes. It is invariant to rotation, translation, scale, being able to capture local geometric and photometric properties. It is particularly useful in the multi-camera environments, where the reconstructed meshes benefit from colour information. Nevertheless, the descriptor is defined generically for any feature available throughout the manifold, colour and curvatures being just some examples. Results in both rigid and non rigid matching tasks are presented. Additionally, the descriptor is integrated within a mesh tracking framework, providing dense matches.

Resumé

Cette thèse propose une méthodologie pour construire un système de reconstruction spatiale et temporelle à partir de plusieurs caméras: étalonnage des caméras et reconstruction 3-D éparse, reconstruction 3-D dense, reconstruction temporelle éparse et dense. Tout d'abord, une formulation probabiliste est développée en association avec des algorithmes de factorisation affine (méthode de reconstruction 3-D fondée sur la factorisation matricielle). Elle permet de récupérer à la fois les paramètres extrinsèques des caméras et les coordonnées 3-D des points de contrôle, étant données les correspondances 2-D de leurs projections et les paramètres intrinsèques des caméras. Le cadre proposé est robuste au bruit et se compare favorablement avec l'ajustement de faisceaux, une méthode standard de minimisation non-linéaire, qui exige un bon estimé initial non loin de l'optimum. Deuxièmement, une méthode d'évolution de maillages est proposée. Elle est capable de gérer les changements topologiques et les auto-intersections sans imposer de contraintes d'échantillonnage sur le maillage. La géométrie exacte du maillage est préservée, à l'exception des parties qui s'auto-intersectent, que l'on retriangularise localement. Des applications sont présentées: le morphing des maillages et la reconstruction 3-D à partir de plusieurs caméras en utilisant des méthodes variationnelles. Troisièmement, une méthode de regroupement de caméras qui utilise l'information de la scène est développée, capable de séparer des reconstructions à grande échelle qui consomment beaucoup de mémoire en plusieurs petites tâches indépendantes de reconstructions partielles utilisant moins de ressources. Enfin, un nouveau descripteur en 3 dimensions est proposé, défini sur des maillages triangulaires échantillonné uniformément. Il est invariant à la rotation, la translation, l'échelle, étant en mesure de capturer les informations géométriques et photométriques locales. Il est particulièrement utile dans le cadre multi-caméras, où les maillages reconstruits bénéficient de la couleur / texture. De plus, le descripteur est défini d'une manière générique pour une fonction quelconque tout au long de la surface (i.e. la couleur, la courbure). Des résultats de correspondance rigide et non rigide sont présentés. Finalement, le descripteur est intégré dans un cadre de suivi temporel dense du maillage.

Acknowledgements

First of all, I would like to thank my supervisors, Radu Horaud and Edmond Boyer, for their help and support, making my experience here in Grenoble very unique. Extended thanks go to the present and past members of the Perception team from INRIA Grenoble, who made my stay in the heart of the lab very enjoyable, both from the social and from the research aspects.

I am grateful to the members of the jury, Adrian Hilton, Renaud Keriven, Jean-Philippe Pons, Slobodan Ilic and Marie-Paule Cani, for accepting to read my manuscript and for providing me with feedback, as well as to my colleagues and friends, Cédric, Albert, Regis and Amaël, who accepted to review the earlier drafts of this manuscript.

Very importantly, I would like to thank my wife, Inna, my two moms and the rest of my family, for their love and support and for encouraging me to apply for this Phd programme.

Special thanks go to Anna and to Paolo, for their rare hospitality and friendship, for offering me a great place to stay during my thesis writing and for making sure that I am always properly fed!

I would like to acknowledge the involvement of Cédric Cagniart and Slobodan Ilic in the work presented in Chapter 6.

This research was supported by the VISIONTRAIN RTN-CT-2004-005439 Marie Curie Action within the European Community's Sixth Framework Programme.

Contents

Chapter 1

Introduction

All living creatures use their senses in order to experience and perceive the surrounding environment. Vision constitutes one of the most advanced and complex of the senses. In the primate cortex, more than half of the cells are dedicated to vision processing. A number of disciplines, such as neurophysiology, psychology and computer science, study the processes that underlie visual perception.

Computer vision is the branch of computer science that strives to reproduce the visual capabilities that many living organisms possess: perceiving depth, understanding the scene geometry, discerning, recognizing and categorizing objects, tracking movements, to list just a few. In computer vision applications, the input to the algorithms generally comes from the imaging sensor of digital cameras, typically replicating the primate eye.

1.1 Motivation

In the recent years multiple camera environments became increasingly available. This came as a direct consequence of the price drops of commodity hardware. Synchronized multi-camera environments provide more redundant information, which can be exploited in order to recover and to interpret the scene geometry. A first step required for any further processing is accurate multi-camera calibration. A number of applications were made possible in such scenarios, including realtime 3-D reconstruction from silhouettes, named visual hulls [49]. The low latency time permitted such solutions to be integrated with virtual reality environments, allowing one to provide real-time visual feedback [7].

More accurate 3-D reconstructions can benefit numerous applications, including medical and multimedia. Therefore, 3-D reconstruction schemes that exploit

the photo-consistency across the cameras have been devised [142]. There is a visible time/accuracy trade-off, which makes the need for better and faster algorithms a reality.

Another very powerful application, establishing the link across different spatial reconstructions of the same moving object/scene across time, is surface tracking. It has numerous applications in the motion capture related industries, such as the film and television, 3-D modeling and medical. Surface tracking provides practical solutions, typically using marker-less tracking methods, either model-based [59, 82, 86] or model-free [34, 160], which can be a powerful alternative to the more expensive dedicated VICON systems [1].

1.2 Problematics

In this thesis we take the necessary steps towards developing a computer vision processing pipeline that allows for dense surface tracking using multiple synchronized cameras. We decompose the pipeline into several building blocks. While solutions exist for each individual step, it is the goal of this thesis to advance the state of the art and propose improved methods. We will briefly describe each block, talking about the problems that it poses. Chapter 2 will describe in more detail each of the building blocks.

Calibration of Multiple-Camera Configurations. The camera calibration problem consists of recovering the intrinsic/internal and extrinsic/external parameters. The intrinsic camera parameters characterize the image formation process. In the case of a standard perspective camera model, it consists of focal length, principal point, skew and distortion. The extrinsic parameters describe the camera position and orientation. A large number of calibration algorithms use calibration objects/patterns with known geometry in order to recover the camera parameters. Others use generic scenes, looking for certain geometrical constraints (i.e. lines) or point correspondences. The calibration procedure that works with a generic scene is called *auto-calibration*. In our scenarios, we are interested by setups with multiple cameras. The above mentioned calibration algorithms are devised for one camera. In order to obtain the best calibration results in multi-camera settings, algorithms that estimates jointly all the camera parameters should be used. *Bundle adjustment* [158] is a standard non-linear minimization technique that provides a correct solution, given an initial solution that is close enough to the correct one. Additionally, bundle adjustment does not handle outliers. *Factorization methods*

[1] http://www.vicon.com

[28] can provide initial solutions for bundle adjustment. Nevertheless, they are equally affected by the outlier problem and there are no mathematically sound solutions existent. Random sampling consensus outlier rejection techniques, such as RANSAC [46], do not easily scale up in the current case, where the number of parameters to estimate is large.

3-D Reconstruction. The 3-D Reconstruction problem is defined as finding a 3-D representation of the scene, using the images provided by a multiple camera setup. At this stage, we can assume that the camera calibration is known. A dense 3-D reconstruction captures the underlying surface, which is typically represented as a mesh. There are numerous approaches to multi-view stereo [142] and they will be reviewed in more detail in the next chapter. One of groups of approaches, called *variational* methods, starts from an initial surface and deforms it, minimizing an error functional. One of the difficulties when dealing with such methods is how to properly evolve explicit mesh representations, since topological changes and self-intersections need to be handled. One other issue that occurs when dealing with large reconstruction tasks that become memory prohibitive is how to properly break down the problem into sub-reconstruction tasks that can be solved independently.

Surface Matching. The surface matching problem is defined as establishing correspondences between 3-D representations of the same moving object at different time instances. There exist a number of methods that define interest point descriptors on surfaces, which can be later on used to find a set of sparse matches. Until very recently, the surface geometry alone was the information used in order to devise such descriptors. However, in the multi-view stereo context, the reconstructed meshes benefit from both geometry and texture. This leaves room for better descriptors to be proposed, taking both geometry and photometric information into account. Such a sparse correspondence is the starting point for dense correspondence algorithms, which diffuse the sparse matches throughout, also taking into account other more global criteria.

1.3 Overview

The goal of the current work is to provide a spatial and temporal 3-D reconstruction vision system using multiple cameras. Initially, a set of noisy 2-D point correspondences is obtained, as well as the internal camera parameters. They are

necessary for the next processing step, which recovers *robustly* the camera positions and orientations as well as the 3-D points. Additionally, a initial mesh is obtained from the 3-D points, which is further on deformed by means of a novel mesh evolution method using an existing variational multi-stereo approach, in order to obtain a final dense reconstruction. The proposed algorithm is scalable, being able to deal with a large number of cameras, by using a scene aware camera clustering approach. Next, several dense reconstructions of the same moving object(s) are obtained at different time steps. An initial set of sparse correspondences is established between meshes at different time-steps using a proposed 3-D mesh descriptor, which is later on used to obtain a dense set of correspondences. An overview of the currently proposed approach is presented in Figure 1.1.

In order to obtain such a complex system, the thesis proposes a number of contributions that serve as building blocks. They will be briefly mentioned in Section 1.4, further detailed in Section 2.4 and explored in depth in the following chapters.

1.4 Organization

In Chapter 2 we will discuss in further detail about the problems one needs to solve in order to obtain a multiple camera system capable of performing spatial and temporal 3-D reconstruction, establishing the links between the proposed contributions. The main goal of this chapter is not to cover in-depth literature review, but is to provide a coherent scenario that introduces the problems and places the proposed contributions in the proper context. Each of the following chapters will introduce in more detail the related work.

Chapter 4 proposes a sparse reconstruction technique using a bayesian formulation that works in conjunction with any factorization method and makes it robust to outliers. It provides both extrinsic camera calibration and sparse point based 3-D reconstruction, given intrinsic camera parameters and 2-D point matches. It provides results comparable with bundle-adjustment, while being a linear method.

Chapter 5 proposes a provably correct geometric approach to explicit surface evolution using triangular meshes. Applications to 3-D reconstruction and mesh morphing are demonstrated.

Chapter 6 proposes a content-aware camera clustering algorithm, able to separate large multi-camera reconstruction problems into separately independent tasks.

Chapter 7 introduces a novel 3-D mesh descriptor that takes into account both geometry and photometric information, available when dealing with meshes obtained via 3-D reconstruction algorithms in the context of multi-view stereo. It is

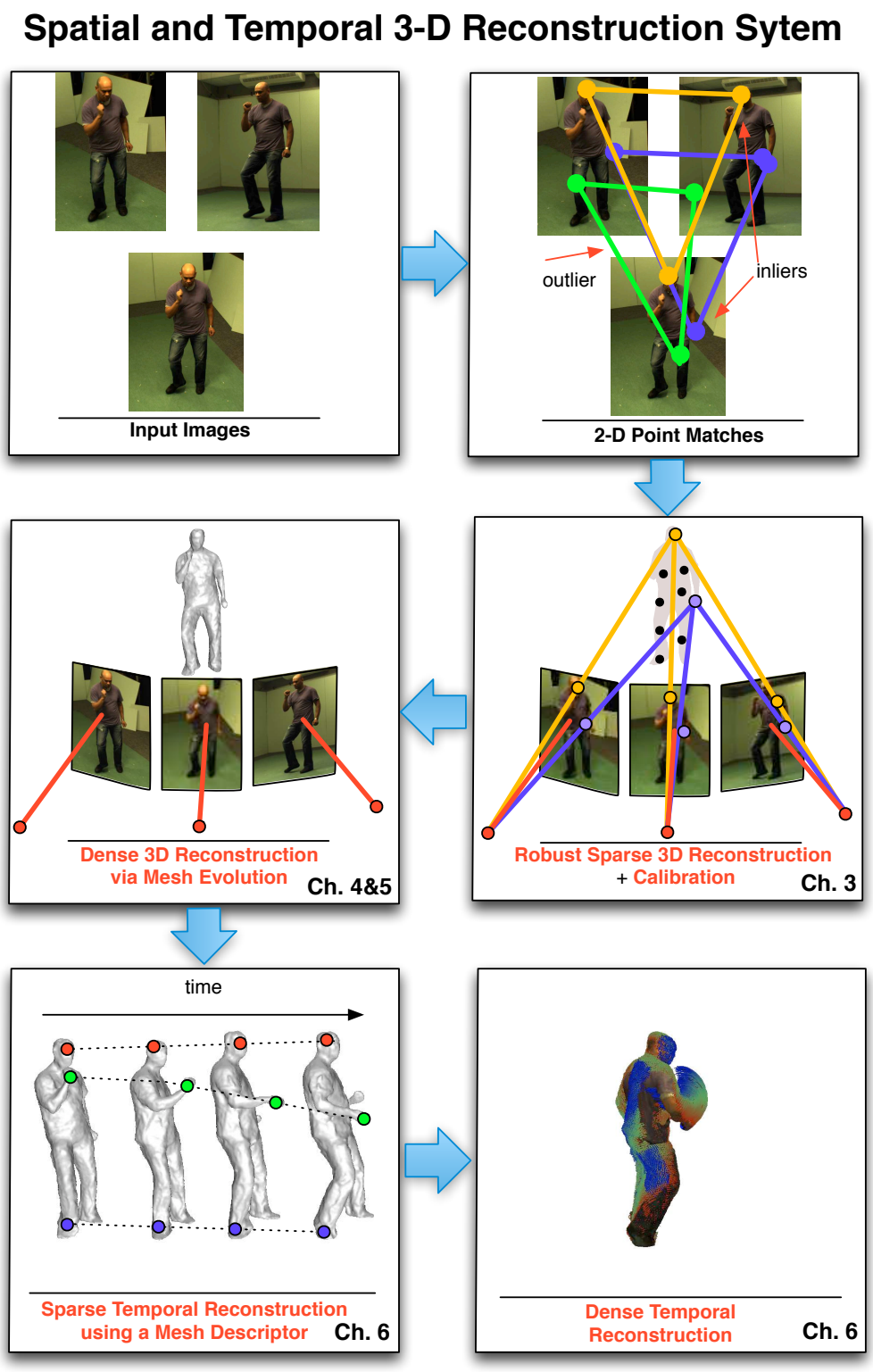


Figure 1.1: Overview of the proposed Spatial and Temporal Multiple-Camera 3-D Reconstruction Pipeline proposed in the current work.

used in order to perform sparse mesh matching. Additionally, the sparse matches are integrated within a dense temporal matching method.

Chapter 8 will conclude the manuscript with a discussion about future work.

Additionally, a complete list of publications is presented in Appendix B.

Chapter 2

Background

In this chapter we will introduce the challenges that one has to deal with when devising a multi-view stereo based 3-D reconstruction and shape matching framework. We will discuss about the constituting topics: multiple camera calibration, 3-D reconstruction and shape matching. The main goal of this chapter is not to cover in-depth literature review, but is to provide a coherent scenario that introduces the problems and places the 4 proposed contributions in the proper context. Each contribution will be further on discussed in more detail in its own chapter, which will also include a broader related work section.

2.1 Calibration of Multiple-Camera Configurations

Camera calibration is one of the fundamental problems of computer vision and tremendous effort has been dedicated to solving it. It consists of finding the relationship that describes how a 3-D point in the world projects into the image plane. This relationship is also called the *forward projection*. Alternatively, the *backward projection* is the relation that, given a projected image point, provides the set of possible 3-D points that map to it.

Camera models vary, starting from the *affine* and *perspective* camera models, ending all the way up to *generic* camera models. The list is rather impressive [132]: perspective, affine (a linearization of the perspective model, including orthographic, weak-perspective, para-perspective), pushbroom (a one dimensional camera), crossed-slit, fisheye (cameras with very large field of view), catadioptric (perspective cameras with mirrors, which can be planar, conical, spherical, parabolic, elliptical, hyperbolic), oblique, radially symmetric, 1D radial, rational

function and generic. As the camera models become more generic, they can account for more variability, making fewer assumption (i.e. in the most generic case, one can consider arbitrary projection rays).

2.1.1 Perspective Camera Model

In the current work we will consider the perspective camera model, due to its simplicity and accuracy with which it can explain the image formation process on most of the commodity hardware digital cameras and lenses available. Following, we shall briefly review perspective camera model. Let us consider the world coordinate system given by (X_W, Y_W, Z_W), which is typically different from the camera coordinate system, denoted by (X_C, Y_C, Z_C). A point $\boldsymbol{P} = [XYZ]^\top$ in word coordinates will project to pixel $p = (u, v)$, with:

$$\begin{bmatrix} \alpha u \\ \alpha v \\ \alpha \end{bmatrix} = \underbrace{\begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_{3\times3}} \underbrace{\begin{bmatrix} \mathbf{R} & \boldsymbol{t} \end{bmatrix}}_{\mathbf{E}_{3\times4}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

where (u_0, v_0) is the projection of the optical center into the image, also named the principal point, expressed in pixel coordinates and f is the focal length. In practice the pixels are not perfect squares, therefore we have two measures f_u and f_v as the two focal lengths, measured in terms of the unit lenths along u and v directions.

The matrix $\mathbf{M}_{3\times4} = \mathbf{KE}$ is called a calibration matrix. It contains both the intrinsic parameters **K**, describing the internal camera geometry, as well as the extrinsic parameters **E**, incorporating the change of coordinate system between the world and the camera. This represents a simplified perspective model that does not incorporate skew and distortion. In most modern cameras the pixels are square, but that does not need be the case. Skew models this, by introducing an additional variable in the intrinsic calibration matrix. Additionally, in an ideal camera there would be no distortion, meaning that each ray would project unaltered. In practice, however, the camera optics, i.e. lenses, introduce both geometric and chromatic aberrations. The geometric distortions represent deviations from the rectilinear projection model, dictating that straight lines in the scene are preserved in the image. The most common geometric aberations are the radial distortion, which can be classified into barrel and pincushion distortion, depending on the effect generated. *Vignetting* is the most predominant chromatic distortion and represents

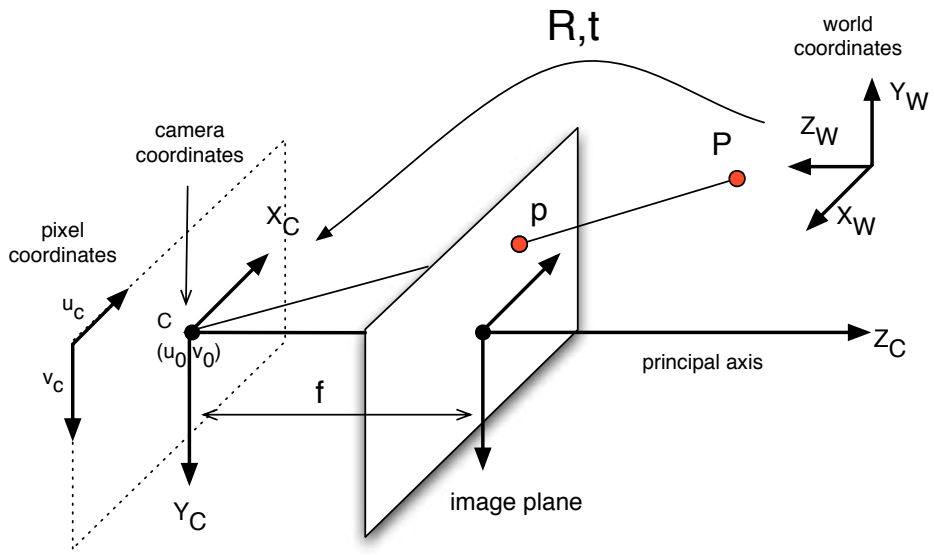


Figure 2.1: Perspective Camera Model

a reduction in the image brightness or saturation at the periphery, compared with the image center.

2.1.2 Affine Camera Models

Nevertheless, one has to keep in mind that the projection of a 3-D point in the camera is a non linear operation. This becomes apparent when regarding the equation (2.1), where, in order to obtain the pixel coordinates (u, v), one has to divide by the projective depth α. This non-linearity makes certain estimation problems harder (i.e. factorization methods, see Section 2.2.1). Therefore, a linearization of the perspective camera model is desirable.

A number of linearized camera models have been proposed: orthographic, weak perspective and para-perspective [69]. An example of such camera projection models, as well as how they compare with the perspective camera model, can be viewed in Figure 2.2. The reader has to keep in mind that such simplified camera models do not hold in cases with strong perspective distortion. Affine camera models are typically used as an intermediate step, to obtain an initial solution. They are generally later on improved and upgraded to a full perspective camera representation.

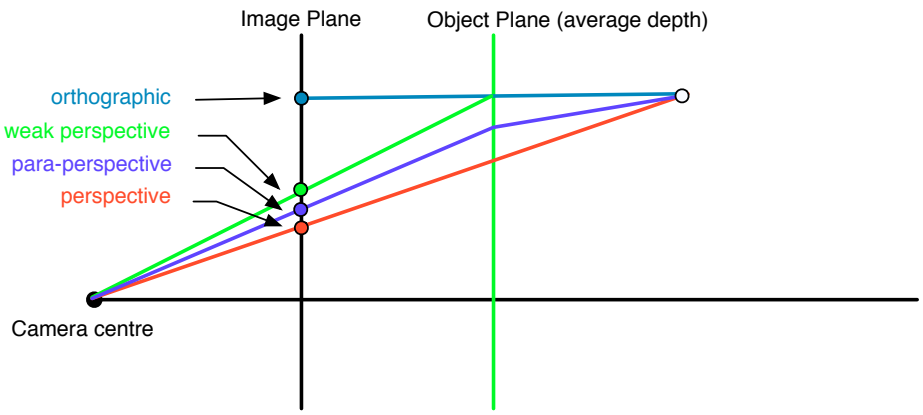


Figure 2.2: Affine / Perspective Camera Models

2.1.3 Single Camera Calibration

There exist numerous methods able to recover the intrinsic parameters of a perspective camera either from 3-D reference objects [45], 2-D planar objects [175], 1-D objects [176] or self-calibration, e.g., from point-correspondences [98], [69], [99].

2.1.4 Multiple Camera Calibration

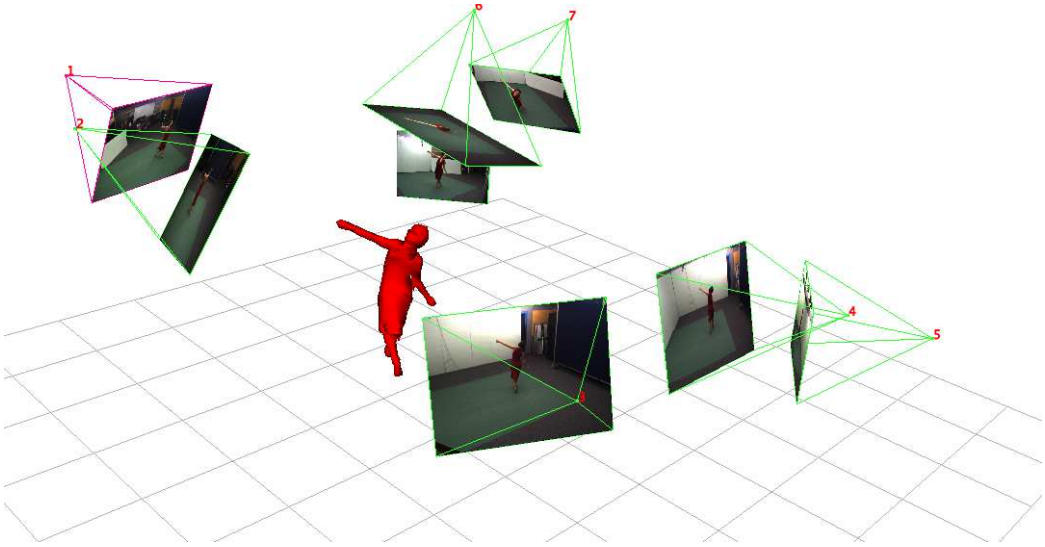


Figure 2.3: A Typical Multi-Camera Environment

As mentioned earlier, we are situating ourselves within the multi-camera environments, as depicted in Figure 2.3. Estimating the camera parameters individually for each camera using the techniques briefly mentioned in section 2.1.3 will produce poorer results than estimating the parameters simultaneously for all the cameras. In the process of doing so, both the camera calibration and the 3-D control points will be recovered. It is for this reason that we have decided to include the discussion about this in Section 2.2.1, dealing with sparse 3-D reconstruction.

2.2 3-D Reconstruction

3-D reconstruction is probably the most central computer vision problem. It consists of trying to recover a 3-D scene, given a number of images. It is in many aspects the inverse problem of *rendering*, which is defined in the graphics community as the process of generating a synthetic image from known 3-D models, textures and lighting by the means of computer programs.

The 3-D reconstruction from multiple camera problem is typically regarded as a two-step process: first, a step of sparse points is reconstructed; then, a dense reconstruction method is initialized from these points and further optimized. Each of the two steps will be further elaborated below.

2.2.1 Sparse 3-D Reconstruction

In most cases, **2-D point correspondences** (of unknown 3-D points) between different cameras are required in order to solve jointly for the camera parameters and the 3-D points. This would correspond to the points $\boldsymbol{s}_{11}$, $\boldsymbol{s}_{12}$ and $\boldsymbol{s}_{13}$, all being a projection of the point $\boldsymbol{P}_1$ in the sample Figure 2.4. In the case of camera calibration, the 2-D point matching problem can be simple, since calibration objects with known geometry are being used [45]. Alternatively, in the general case with no scene priors, affine invariant image descriptors, such as SIFT (Scale Invariant Feature Transform) [97] or MSER (Maximally Stable Extremal Regions) [108], are chosen for wide baseline matching, or, in the case of nearby cameras, an image tracker [21]. Similarity measures or other stereo matching techniques [139] can also be used to provide potential matches.

Given the 2-D correspondences, there are two paradigms for obtaining the 3-D reconstruction and the calibration: *global methods* and *local methods*. Global methods use all the available correspondences at once, attempting to find a solution that minimizes a joint reprojection error. Local methods attempt to build

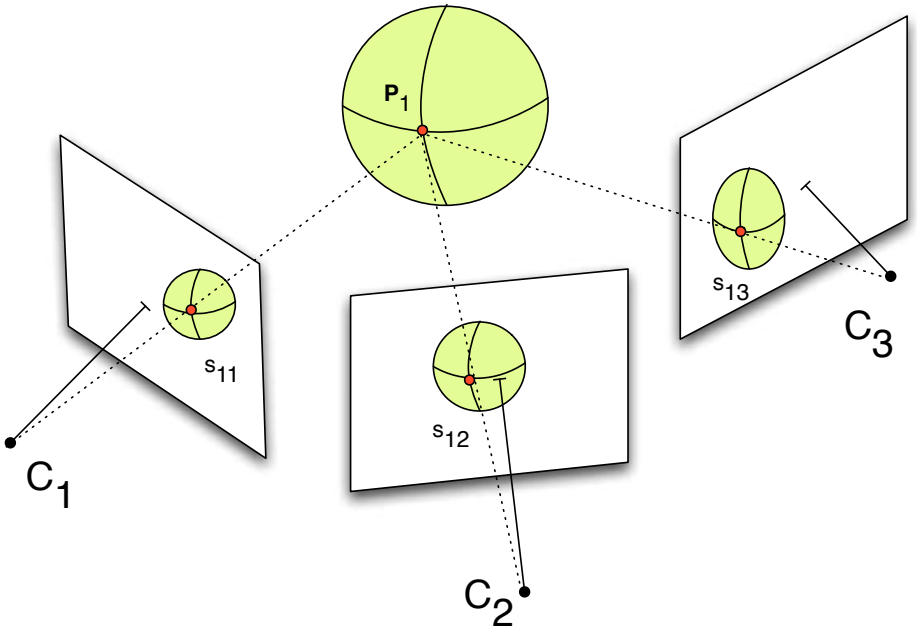


Figure 2.4: Point projections in a multiple camera setup. In this example the point $\boldsymbol{P}_1$ projects in cameras C_1, C_2 and C_3 at coordinates $\boldsymbol{s}_{11}$, $\boldsymbol{s}_{12}$ and $\boldsymbol{s}_{13}$, respectively.

a reconstruction gradually, by typically considering pairs or triplets of cameras which are incrementally merged into a global solution.

Global Methods

Given the 2-D correspondences, *bundle adjustment* provides both a general method and practical algorithms for solving this reconstruction problem using maximum likelihood [158]. Nevertheless, bundle adjustment is non-linear in nature and sophisticated optimization techniques are necessary, which in turn require proper initialization. There have been attempts [27] to incrementally build the solution using bundle-adjustment, adding one camera at the time.

Consider that we have n 3-D points $\boldsymbol{P}_i, \forall i \in [1, n]$ and c cameras with the 3×4 perspective projection matrices $\mathbf{M}_j, \forall j \in [1, c]$. The points $\boldsymbol{P}_i$ will project at coordinates $\hat{\boldsymbol{s}}_{ij}$, where $\hat{\boldsymbol{s}}_{ij} = (\hat{u}_{ij}, \hat{v}_{ij})$ and $(\hat{u}_{ij}, \hat{v}_{ij})$ is obtained using the perspective projection equation (2.1). Such a scenario is depicted in Figure 2.4. Now, consider that we are only given $\boldsymbol{s}_{ij}$ point correspondences, and we are searching for $\mathbf{M}_j$ and $\boldsymbol{P}_i$. This can be formulated as a minimization of the reprojection error:

$$\min_{M_i,P_j} \sum_{i,j} ||\hat{\boldsymbol{s}}_{ij} - \boldsymbol{s}_{ij}||^2 \tag{2.2}$$

Affine Factorization Methods provide a practical way obtain an initial reconstruction, which can be further on refined to a perspective camera model using bundle adjustment. The first practical solution was provided by Tomasi and Kanade [156] in 1992, in the case of complete data (each 3-D point will have correspondences in all images) under an affine camera model. They observed that, if all the correspondences are stacked in a data measurement matrix **S**, then **S** can be written as $\mathbf{S} = \mathbf{MP}$, where **M** is the motion matrix representing the cameras, and **P** represents the 3-D points. Their solution is based on a singular value decomposition of the **S** matrix, which in practice solves for the minimum 2-D reprojection error problem:

$$\min_{\mathbf{M},\mathbf{P}} ||\mathbf{S} - \mathbf{MP}||_F$$

Since then, numerous other factorization methods have been proposed, being able to deal with issues such as missing data and uncertainty [1, 29, 62, 13, 155].

Nevertheless, one has to keep in mind that all affine factorization methods assume a linearized camera model (i.e. orthographic, weak perspective, paraperspective) in order to be able to decompose the measurement matrix **S** into a motion matrix **M** and the 3-D points **P**.

This is what motivated the development of factorization frameworks that can perform affine to perspective upgrades [32]. Alternatively, numerous factorization methods in the projective space have been devised, followed by Euclidean upgrade [152, 159, 100, 101, 104, 129, 122]. None of the above-enumerated methods, including bundle adjustment, deal intrinsically with outliers and thus, can be greatly affected by noise in the measurements, since they are trying to minimize the reprojection error.

RANSAC [46] has been typically used in robust estimation problems in order to draw random samples from the available data and provide sub-solutions. While this can be a very effective way to prune outliers, the process becomes very time consuming when the number of minimum samples required to solve for all the free parameters becomes large. This is the case when dealing with enough multiple cameras. As we will see in the next section, when dealing with smaller problems, RANSAC is a very effective method.

Alternatively, robustness may be achieved through *adaptive weighting*, i.e., by iteratively updating a weight matrix **W** associated with the data, such as it is done by Aanaes et al. [1]. Their method uses a robust loss function (see Stewart [150]

and Meer [111] for details) to iteratively get a temporary optimum. Nevertheless, the convergence properties of combining iterative factorization algorithms with robust estimators are not fully understood. Additionally, the robust estimator tuning parameter (i.e. the cut-off value) is typically chosen by hand.

Local Methods

The philosophy behind the local methods is to incrementally build a global reconstruction by integrating smaller partial results. At first, pairs of cameras are considered. Given at least 8 matches between the two cameras, the epipolar constraint can be recovered [69], relating the two cameras. The epipolar constraint is a 3×3 rank 2 matrix, called the *fundamental matrix*, that relates the match $\boldsymbol{p}_1$ from the first camera to the match $\boldsymbol{p}_2$ from the second camera via:

$$\boldsymbol{p}_1^\top \mathbf{F} \boldsymbol{p}_2 = 0 \tag{2.3}$$

where homogeneous 2-D coordinates have been used, i.e. $\boldsymbol{p}^\top = [x\ y\ 1]$. The fundamental matrix can be robustly estimated using RANSAC [46], due to the small number of model parameters (i.e. 8). In practice, the fundamental matrix constrains the match of a point $\boldsymbol{p}_1$ in the first image to a line in the second image. Such a step was employed in [27].

Further on, in a similar spirit, a *trifocal tensor* [69] can be estimated for triplets of cameras. The trifocal tensor consists of three 3×3 matrices with 18 independent degrees of freedom. It represents for triplets of cameras what the fundamental matrix represents for camera pairs. The tensor can also be robustly estimated using RANSAC, thus allowing further pruning of the false matches. Such a method was employed in [105].

Using subsets of cameras, partial reconstructions can be recovered, up to a scale. The partial reconstructions can be stitched together using control points that occur across different sets of partial reconstructions [129, 107], or using the cameras, if the partial reconstructions incorporate at least 4 cameras [106]. Bundle adjustment can be used incrementally to refine each newly added partial reconstruction [27] or as a final step.

The method proposed by [107] is particularly attractive, being able to deal with large amounts of missing data, large measurement matrices and outliers (for more details, see [103]). Impressive reconstruction results are shown, specially for the *city paper model* dataset, with 2126 images, 660037 reconstructed 3-D points and 99.85% missing data. The method starts from pairwise perspective reconstructions. Then, given the pairwise relative rotations, global camera rotations are

estimated linearly in least squares. Camera translations are then estimation using an existing convex optimization scheme, which guarantees a global optimum [81]. Robustness is achieved by keeping minimal point configurations for the camera pairs (four points chosen), sufficient to represent a pairwise reconstruction, while providing a important computation speedup. In addition, an iterative removal of pairwise reconstructions with the largest residual and re-registration removes most of non-existent epipolar geometries.

Global methods, benefitting from a clean formulation, attempt to obtain a solution that takes into account all data measurements. However, given challenging noisy configurations, they can diverge. Alternatively, local methods attempt to progressively obtain more complete reconstruction, having the great advantage of recovering gracefully, even if local reconstructions fail. Nevertheless, they can diverge in the merging step. Other algorithms, such as [107], can be thought of as a hybrid approach between the local and global methods. The proposed method takes local decisions in obtaining pairwise reconstructions and in choosing representative points robustly, but solves globally for the camera rotations and translations.

If we consider that the camera calibration is available [53, 54], the 3-D sparse reconstruction problem becomes easier. Once the false matches have been eliminated using the epipolar constraints the 3-D points can be easily recovered. Given the projections of a 3-D point into two known cameras, the 3-D coordinate of the point can be recovered via the process known as triangulation.

2.2.2 Dense 3-D Reconstruction

3-D points are just one of the possible **shape representations**. We will review the most popular shape representations, talking about the advantages and disadvantages of each. A shape can be represented as a mesh, a point cloud, a collection of depth maps, the level set of an implicit function or as an occupancy grid [56]. We will then review at a very high level the 3-D reconstruction algorithms.

Shape Representation

Point Clouds can be used to describe a shape. The Factorization algorithms described earlier can provide such a representation. They do not provide exact information about the shape boundary information. In order to derive **Oriented Point Clouds**, local shape normal orientations can be inferred by local surface fitting using the neighouring points [127, 39] or taking into account camera visibility and texture information [54]. Nevertheless, such a representation provides only

a sampling of a continuous shape representation, which can be discretely represented either explicitly using *meshes*, or implicitly using *level sets* or *occupancy grids*.

It is possible to construct a **continuous shape representation from point clouds**, either oriented or non-oriented, under the assumptions that there are enough point samples to unambiguously describe the surface and that the surface is locally smooth [20, 12, 83, 90, 8, 128]. We shall detail each of the above mentioned approaches, since they provide the link between the sparse and dense 3-D reconstruction methods, a key step in the proposed 3-D reconstruction and matching pipeline. Such methods provide an initial mesh surface, which can be further on deformed such that it minimizes an energy functional. The first algorithm that has been proposed with theoretical guarantees in reconstruction, named *Crust*, is due to Amenta et al. [10]. This algorithm takes advantage of the structure of the Voronoi diagram of the input point set in order to reconstruct the surface. Additionally, Amenta et al. [12] proposed an improved version, named *Power Crust*, based on 3-D Delaunay triangulations, Voronoi cells and the medial axis approximation. Yet another improved version, named *Cocone*, was proposed by Amenta et al. in [11]. Boissonnat et al. [20] derive a method to compute smooth surfaces of arbitrary topology from unorganized 3-D oriented points via natural neighbour interpolation of distance functions. In related work, Petitjean et al. [128] propose a reconstruction method from unoriented points based on regular interpolants, which are polygonal approximations of curves and surfaces satisfying a new regularity condition. Kazhdan et al. [83] propose another algorithm to recover a surface from oriented points, casting it as a spatial Poisson problem. Alliez et al. [8] propose a method that recovers watertight surfaces from unoriented point sets using the Voronoi diagram of the input point set and deducing a tensor field whose principal axes and eccentricities locally represent the most likely direction of the normal to the surface together with the confidence in this direction.

Additional constraints can be taken into account if the camera geometry is known. Labatut et al. [90] propose a space carving approach using graphcuts in order to eliminate from a triangulation of the space the tetrahedra that violate the camera visibility constraints (the tetrahedra that lie between the 3-D points and the cameras). In related work, the visual hull methods exploit the known camera geometry as well as the silhouette information in order to find the surface that resides inside the silhouette cone intersections [49, 92].

Meshes are one of the most widely used surface representations, specially in computer graphics. A mesh is defined by its vertices (3-D points), together with the connectivity information that defines the facets and edges. A facet can be any polygon, but typically triangular meshes are among the most widely used. That is

due to the simplicity of the representation and due to the availability of graphics hardware that can render triangles lists with very low latency, to name just a few. Meshes provide a compact representation, allowing for an adaptive resolution. Differential geometry properties, such as normal and curvature [38, 40] have been defined. When using meshes as a representation in the context of multi-view 3-D reconstruction, a considerable number of approaches start from an initial surface, which they later on deform such that it reduces an error functional (the so called variational methods). Performing surface evolution using meshes it is a difficult problem, due to the two main issues that need to be taken into account: self-intersections and topology changes. In 2000, McInerney and Terzopoulos [109] introduced topology adaptive deformable curves and meshes, called T-snakes and T-surfaces. However, in solving the intersection problem, the authors use a spatial grid, thus imposing a fixed spatial resolution. In addition, only offsetting motions, i.e. inflating or deflating, were allowed. It is only very recently (2003) that a first heuristic solution was proposed by Lachaud et al. [91]. Nevertheless, the proposed solution requires equal sampling throughout the mesh (all the triangles to be of approximately equal size). An elegant alternative method was also proposed in 2007 by Pons et al. [130], using tetrahedral meshes to represent the shape instead of triangular meshes, together with a restricted 3-D Delaunay triangulation. However, being a Delaunay based method, it will be affected by undersampling.

Level Sets, originally introduced by Osher and Sethian [125], propose a continuous shape representation as the zero level set of an implicit function. The function is negative inside the shape, positive outside, and zero where the surface resides. The zero level set is also called the interface. Moving the shape reduces to changing the values of the implicit function. Such a representation overcomes the typical topological problems encountered when dealing with explicit representations. Additionally, due to the fact that a regular grid is typically used to represent the implicit function, standard finite difference methods can be applied [123]. The explicit surface has to be extracted at each iteration from the implicit function using the marching cubes algorithm [94]. Some of the drawbacks of such a representation are the fact that the original 2-D manifold (the surface) has been embedded in a higher dimension (3-D), thus requiring more memory in order to store it. Narrow band representations alleviate this problem [2]. Additionally, level sets are not suitable to track interface properties, such as texture coordinates.

Occupancy Grids are representations that partition the space into cells. Each cell can either be inside the shape (thus occupied) or outside (thus empty). Standard voxel grids [9, 50, 58] or tetrahedra [19] can be used.

Depth Maps are representations associated with one image, typically obtained from dense two-frame stereo algorithms. For a review of the evaluation of such

algorithms, please see [139] and [1]. Each pixel value represents the depth at which a surface is visible for that particular pixel. Such a representation is not complete, since it cannot represent closed surfaces. There are several methods that, in the context of multi-view stereo, fuse multiple depth-maps in order to obtain a robust representation [57, 168, 112, 169].

3D Reconstruction

There exist a considerable number of 3-D reconstruction methods that have been developed recently. For a comparison and evaluation of different multi-view stereo approaches, please consult [142] and [2]. They are categorized in roughly 4 categories. The *first* category operates by computing a cost function on a 3-D volume, which is later on used to extract a surface from it. It includes the voxel colouring algorithms [9] and graph cut algorithms [163]. The *second* class includes the variational methods and it works by iteratively evolving a surface to decrease or minimize an error function [131]. Space carving methods [89] progressively remove inconsistent voxels. The *third* class of methods combine multiple depth maps, ensuring a consistent 3-D scene [168, 112, 169]. Finally, the *fourth* set of algorithms [52, 129, 54] take a two step approach: firstly, they extract a set of keypoints that are robustly matched across images; secondly, they fit a surface to the reconstructed oriented points. The most popular methods [20, 12, 83, 8, 90, 128] for performing the second step have been reviewed in the earlier subsection.

An issue of interest that naturally arises in this field is how to efficiently deal with scenarios where there are lot of images and, due to memory requirements, they cannot all be processed at the same time.

2.3 Surface Matching

So far we have focused on the necessary steps required to devise a workflow for obtaining 3-D mesh reconstructions using multiple synchronized cameras. Given such reconstructions at different time-steps, the natural question that follows is how can a temporal correspondence be established between the separate surfaces. In the same spirit as in the 3-D reconstruction framework, this is a two-step process: first establish a sparse set of correspondences between keypoints on the

[1] http://vision.middlebury.edu/stereo/
[2] http://vision.middlebury.edu/mview/

two meshes; then, densify the matches in order to obtain a one-to-one correspondence between the two surfaces for each of the constituting vertices. Numerous applications can benefit from the temporal correspondence: mesh reconstructions can be improved to be temporally coherent; the texture information can be compressed and optimized across the whole sequence; motion segmentation can be easily performed on the vertex tracks, leading to automatic motion segmentation and possibly automatic skeletal extraction, in the case of piece-wise rigid motions.

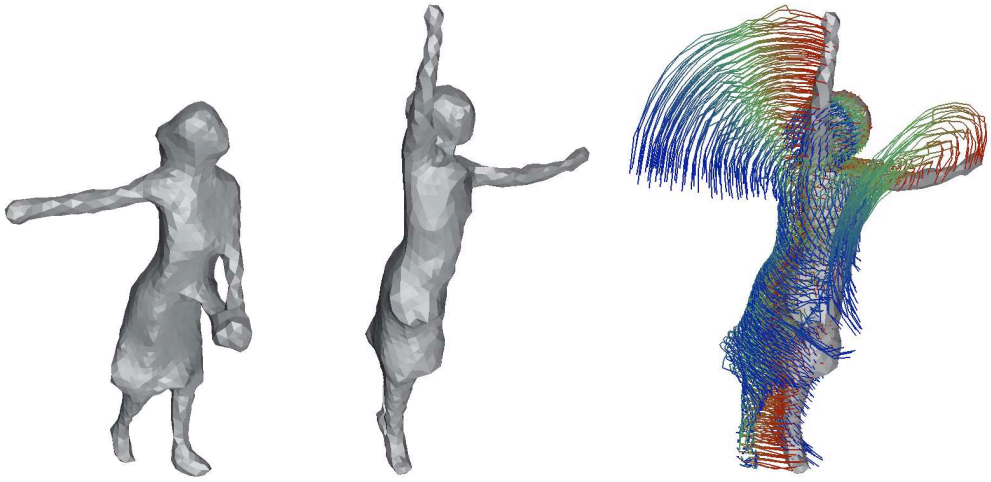
2.3.1 Sparse Surface Matching

A number of surface shape descriptors have been been proposed, describing either the local or the global geometry. Among some of the most successful 3-D surface descriptor, we can enumerate the 3-D Spin Images [78] and 3-D Shape Contexts [88, 51]. These are descriptors that rely solely on the surface geometry. For detailed literature surveys, please consult [154, 30]. However, when dealing with meshes in the multi-view stereo context, colour/texture information is also available. On the negative side, the 3-D reconstructions from multiple images are noisier, thus more challenging, than the surfaces used in the graphics community to perform mesh matching [25]. Often the recovered topology is not necessarily the correct one and often it cannot be known in advance. Imagine a person, with his/her arm touching the body, as depicted in Figure 2.5. There has been very little work [167] in trying to devise a mesh descriptor that incorporates photometric and geometric information.

2.3.2 Dense Surface Matching

In this section we discuss the problem of capturing the evolution of a moving and deforming surface, in particular moving human bodies, given multiple videos. A large variety of directions can be followed, depending on the *a priori* knowledge of the observed shape, on the representation chosen for surfaces and on the information taken into account for deformations. *Model-based approaches* assume a known model of the observed surface, which is tracked over time sequences, hence solving for time correspondences. This model can be locally rigid, e.g [59, 82, 31], or deformable, e.g. [35, 138]. Unfortunately, exact models need to be available, which is seldom the case in general situations. In particular, the topology of the surface can evolve over time as shown in Figure 2.5. As a consequence, approaches in this category are restricted to specific scenarios.

In contrast, non model-based approaches try to find displacement fields between 2 different instants in the sequence. In this category, *scene flow approaches*



(a) (b) (c)

Figure 2.5: An example of a surface for which the topology can hardly be known in advance. The belt of the dress forms a new protrusion that appears and disappears (a)-(b). (c) Dense point trajectories computed from (a) to (b) using our proposed method from [160].

consider dense vector fields with various representations including voxels [161, 119], implicit representations [131] or meshes [34]. However, the associated differential methods are limited to small displacements between successive frames. Alternatively, *feature-based approaches* [14, 26, 148] consider meshes and allow for larger motions by casting the problem as a labeling between 2 meshes using local geometric or photometric information. This labeling solves for partial correspondences between 2 frames only and might lose efficiency when applied over long sequences, in particular as topological changes occur.

An interesting related research direction has been proposed in [4], where a spatio-temporal reconstruction (thus 4-D) has been obtained from exiting individual visual hulls at different time-steps. The 3-D reconstruction at a given time instant can be obtained by intersecting this 4-D mesh representation with a temporal plane, thus enabling the interpolation of scene between consecutive available frames.

Our approach, described in Varanasi et al. [160], is grounded on the observation that natural surfaces are usually arbitrarily shaped and difficult to model *a priori*. In addition, shapes can significantly evolve over a time sequence. For instance, human bodies are usually covered by clothes, characterized by changing topology. To handle such deformations, we use meshes which are morphed from one frame to another. Like feature-based approaches, we use photometric cues

provided by images and geometric cues provided by the recovered meshes. However, instead of looking for a dense match between the vertices of the 2 meshes, we use a sparse, but robust set of matches and its associated displacement vector field to obtain a complete transfer function from one mesh to another, with possible topological changes. This approach provides both a consistent surface evolution over time and dense point trajectories on the surface. While the details of this approach will not be detailed in the current work, an overview is provided in Chapter 7.

2.4 Contributions

In the current thesis we propose four contributions. They all integrate as part of the building blocks needed to devise a spatial and temporal reconstruction system from multiple cameras. They are the following:

- **A Robust Factorization Framework for Simultaneous Extrinsic Camera Calibration and Sparse 3-D Reconstruction**: we propose a general method that works in conjunction with any affine factorization method and makes it robust to outliers. We cast the problem in the bayesian framework. We consider a Gaussian/Uniform mixture model, providing a practical solution via the E-M algorithm [17]. This allows us to address the robust parameter estimation problem within a data clustering approach. In addition, we show how such a framework can be further embedded into an iterative perspective factorization scheme. We carry out a large number of experiments to validate our algorithms and to compare them with existing ones. We also compare our approach with another proposed factorization method that uses M-estimators. This work will be detailed in Chapter 4.

- **A Mesh-Based Topology Adaptive Framework for Surface Evolution**: we propose a new fully geometric method, named TransforMesh, for explicit mesh-based surface evolution. It gracefully overcomes the limitations of traditional mesh-based methods, handling self-intersections and topology changes in a natural manner, while not imposing a uniform mesh sampling. The method preserves the geometry of the original mesh, except in the areas that contain self-intersections. We show how such a framework can be used in the context of multi-view 3-D reconstruction, using it in conjunction with the variational method proposed by Pons et al. [131]. Additionally, we present results of using TransforMesh to perform mesh morphing. This work will be detailed in Chapter 5.

- **A Camera Clustering Framework for Part-based 3-D Reconstruction**: we propose a scene aware approach to partition the camera images, either semi-automatic, through clustering, or user guided, via a geometric modeling interface. In order to reduce the volume of image data we need to access simultaneously, we use subsets of the original image set and evolve the parts of the surface corresponding to those images by maximizing the photo-consistency. This work will be detailed in Chapter 6.

- **A Novel 3-D Mesh Descriptor**: we propose a novel 3-D descriptor on triangular meshes that takes into account both surface geometry and photometric information. We name this descriptor MeshHOG (Mesh Histogram of Oriented Gradients). It represents a generalization of the 2-D HOG (Histogram of Oriented Gradients) descriptor [33]. The photometric information is accumulated on each of the mesh vertices using the median colour in the viewable cameras. Additionally, we introduce a MeshHOG detector on triangular meshes. This new descriptor can be used as the new building block for obtaining the initial sparse mesh matches within a dense mesh matching / tracking framework. We will show how the descriptor is successfully integrated in such a mesh tracking paradigm [160]. This work will be detailed in Chapter 7.

Chapter 3

Extended Abstract (French)

Ce résumé en français est un aperçu de la thèse, écrite originalement en anglais. Nous invitons le lecteur à se rapporter au manuscrit anglais pour plus d'informations, résultats, preuves et précisions.

3.1 Introduction

Dans cette thèse nous présentons les étapes nécessaires pour développer un système de vision par l'ordinateur qui permet la reconstruction spatiale et temporelle des surfaces à partir de plusieurs caméras synchronisées. Le pipeline est décomposé en plusieurs blocs. Bien qu'il existe déjà des solutions pour chaque bloc composant, le but de cette thèse est d'avancer l'état d'art et de proposer nouvelles des solutions plus performantes.

Initialement, des correspondances bruitées de points 2-D sont obtenues, ainsi que les paramètres internes des caméras. Ils sont nécessaires pour l'étape suivante, qui retrouve d'une manière robuste les positions 3-D et les orientations des caméras, ainsi que les points 3-D. Cette procédure est détaillée dans la Section 3.2. En plus, un maillage initial est obtenu à partir des points 3-D, qui évolue ultérieurement pour minimiser l'erreur de photo-consistance en utilisant une nouvelle méthode d'évolution de maillage. Cette partie est décrite dans la Section 3.3. La méthode de reconstruction peux gérer un grand nombre de caméras, en proposant une méthode de regroupement de caméras qui permet d'effectuer des calculs localement. L'algorithme de regroupement de caméras est décrit dans la Section 3.4. Par la suite, plusieurs reconstructions du même objet en mouvement sont obtenues aux différents instants. Elles sont mises en correspondance en utilisant un nouveau descripteur 3-D de maillage proposé dans la Section 3.5. Enfin,

les correspondances éparses sont utilisées dans un cadre de suivi de maillage dense pour obtenir des correspondances denses entre le maillage aux différents instants. Un résumé de ce pipeline est illustré dans la Figure 3.1.

3.2 Factorisation probabiliste robuste

La factorisation a été introduite par Tomasi & Kanade [156] comme une solution élégante à la reconstruction affine à partir de plusieurs caméras; leur solution initiale, basée sur la décomposition en valeurs singulières (SVD), utilise un modèle de caméra perspective faible. La méthode a été ultérieurement amélioré par Morris & Kanade [117], Anandan & Irani [13], Hartley & Schaffalitzky [68] et bien d'autres.

Le problème peut être formulé comme une minimisation de la norme de Frobenius suivante:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\mathbf{S} - \hat{\mathbf{S}}(\boldsymbol{\theta})\|_F^2 \tag{3.1}$$

où $\mathbf{S} = [\boldsymbol{s}_{ij}]$ est la matrice de mesure qui contient les correspondances 2-D des observations.

$\hat{\mathbf{S}}(\boldsymbol{\theta}) = \mathbf{MP}$ est la matrice de prédiction qui peut être factorisée comme la matrice *affine* des caméras (aussi appelée la matrice de mouvement) $\mathbf{M}$ et la matrice *affine* des points 3-D (aussi nommée la matrice de forme) $\mathbf{P}$. On dénote par $\boldsymbol{\theta}$ les paramètres de mouvement **et** de forme.

Dans cette section nous avons exprimé le problème de la factorisation robuste dans le cadre EM (maximisation de l'espérance) [48]. Nous séparons le problème de classification des correspondances 2-D observées en deux catégories : données correctes (inlier) et données erronées (outlier). Ainsi, nous modélisons la probabilité d'observations par un mélange Gaussien/Uniforme. Ceci mène à une formulation du maximum de vraisemblance avec les variables cachées qui peuvent être résolues avec l'algorithme de EM [37], [110], [48].

Tout d'abord, une formulation probabiliste est développée en association avec les algorithmes de factorisation affine (méthode de reconstruction 3-D fondée sur la factorisation matricielle); il permet de récupérer à la fois les paramètres extrinsèques des caméras multiples et les coordonnées 3-D des points de contrôle, étant donné les correspondances 2-D de leurs projections dans les images et les paramètres intrinsèques des caméras. Le cadre proposé est robuste au bruit et se compare favorablement avec l'ajustement de faisceaux, une méthode standard de minimisation non-linéaire, qui exige à l'origine une bonne initialisation.

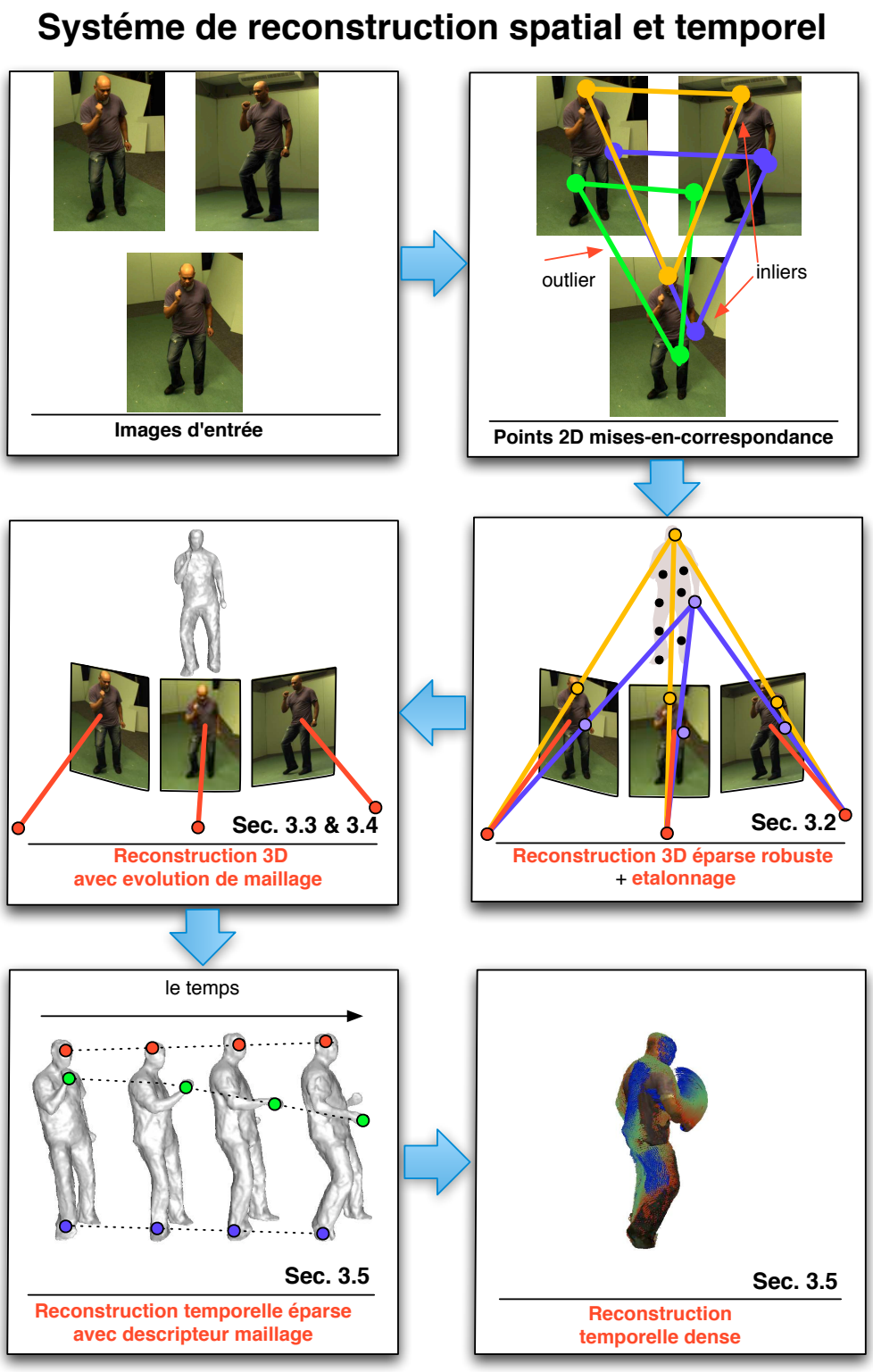


Figure 3.1: Résumé du pipeline proposé dans cette thèse pour réaliser une reconstruction dense spatiale et temporelle à partir des plusieurs caméras.

3.2.1 Modélisation

On considère un ensemble de points dans des images, $\boldsymbol{s}_{ij}$ ($1 \leq i \leq k$, $1 \leq j \leq n$), qui représentent les valeurs observées par un nombre égal de variables aléatoires s_{ij}. On introduit un autre ensemble de variables aléatoires z_{ij}, qui attribuent une valeur à chaque catégorie: *données correctes* et *données bruites*. Plus précisément, $z_{ij} =$ inlier décrit le fait que $\boldsymbol{s}_{ij}$ est correcte et $z_{ij} =$ outlier represente le fait que $\boldsymbol{s}_{ij}$ est une donnée bruite.

On définie les probabilités a priori. On considère que la surface de l'image i est A_i est que toutes les images ont la même surface $A_i = A$. Si une observation est correcte, elle est contenue dans une petite vignette circulaire a de rayon σ_0, $a = \pi\sigma_0^2$. La probabilité a priori d'un inlier est la partie de l'image limitée à la petite vignette circulaire:

$$P(z_{ij} = \text{inlier}) = \frac{a}{A} \tag{3.2}$$

D'une manière similaire, si l'observation est un outlier, la probabilité doit exprimer le fait que le point reste en dehors de la vignette :

$$P(z_{ij} = \text{outlier}) = \frac{A - a}{A} \tag{3.3}$$

De plus, une observation $\boldsymbol{s}_{ij}$, étant donné que c'est un inlier, doit se trouver dans le voisinage d'une estimation $\hat{\boldsymbol{s}}_{ij}$. Donc, si on modélise la probabilité de l'observation $\boldsymbol{s}_{ij}$ *étant donné que c'est un inlier* par une gaussienne centré sur $\hat{\boldsymbol{s}}_{ij}$ et de matrice de covariance $\mathbf{C}$ de taille 2×2, on obtient:

$$P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier}) = \frac{1}{2\pi(\det \mathbf{C})^{1/2}} \exp\left(-\frac{1}{2} d^2_{\mathbf{C}}(\boldsymbol{s}_{ij}, \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\right) \tag{3.4}$$

où $d_{\mathbf{C}}$ est la distance Mahanalobis.

$$d^2_{\mathbf{C}}(\boldsymbol{s}_{ij}, \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) = (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{3.5}$$

Quand l'observation est un outlier, elle peut se trouver partout dans l'image. Donc, on modélise la probabilité d'une observation $\boldsymbol{s}_{ij}$ *etant donné que c'est un outlier* par une distribution uniforme sur l'image:

$$P(\boldsymbol{s}_{ij}|z_{ij} = \text{outlier}) = \frac{1}{A} \tag{3.6}$$

En appliquant la loi de Bayes et après quelques simplifications (i.e. $a << A$), la probabilité a posteriori d'être un inlier α_{ij}^{in} est:

$$\alpha_{ij}^{in} = P_{\boldsymbol{\theta}}(z_{ij} = \text{inlier}|\boldsymbol{s}_{ij}) = \frac{1}{1 + \frac{2}{\sigma_0^2}(\det \mathbf{C})^{1/2} \exp\left(\frac{1}{2} d^2_{\mathbf{C}}(\boldsymbol{s}_{ij}, \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\right)} \tag{3.7}$$

Si on considère que la matrice de covariance est isotropique, $\mathbf{C} = \sigma^2 \mathbf{I}_2$, (3.7) devient:

$$\alpha_{ij}^{in} = P_{\boldsymbol{\theta}}(z_{ij} = \text{inlier}|\boldsymbol{s}_{ij}) = \frac{1}{1 + \frac{2\sigma^2}{\sigma_0^2} \exp\left(\frac{\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})\|^2}{2\sigma^2}\right)} \tag{3.8}$$

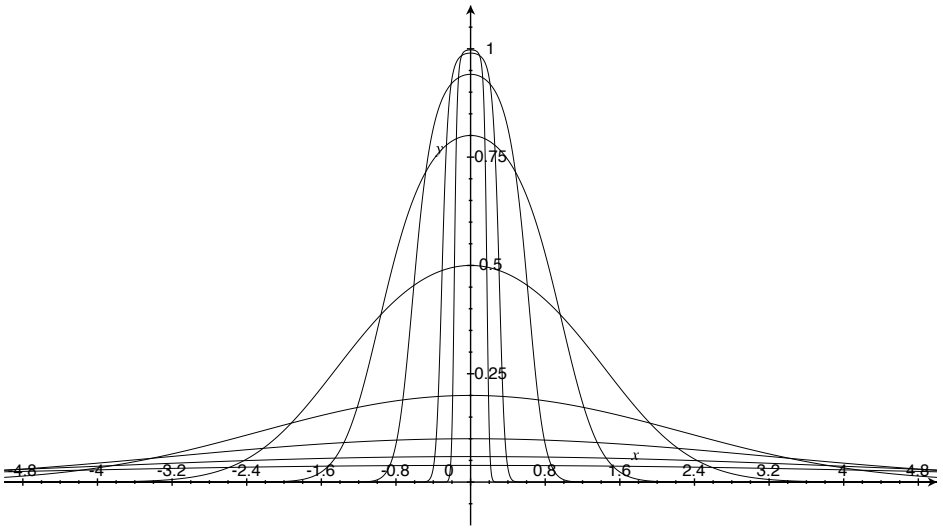


Figure 3.2: Des graphes pour la probabilité a posteriori d'une observation d'être un inlier, i.e., $f_\sigma(x) = 1/(1 + \sigma^2 \exp(x^2/2\sigma^2))$. Cette fonction correspond à eq. (3.8) avec $\sigma_0^2 = 2$. Lorsque la variance baisse, i.e., $\sigma = 5, 4, 3, 2, 1, 0.5, 0.25, 0.1, 0.05$, la fonction devient de plus en plus discriminante.

Maximum de vraisemblance avec des inliers

Le maximum de vraisemblance maximise le logarithme de la vraisemblance jointe de toutes les mesures $P_{\boldsymbol{\theta}}(\mathbf{S})$, en considérant que toutes les mesures sont distribuées de manière indépendante et identique:

$$P_{\boldsymbol{\theta}}(\mathbf{S}) = \prod_{i,j} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}) = \prod_{i,j} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier}) \tag{3.9}$$

Le logarithme de la vraisemblance devient:

$$Q_{ML} = \frac{1}{2}\sum_{i,j}\Big((\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) + \log(\det \mathbf{C})\Big) \tag{3.10}$$

Les paramètres des caméras et les points 3-D peuvent être estimées en minimisant $\boldsymbol{\theta}$ suivant:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{3.11}$$

Dès qu'une solution optimale a été trouvée, i.e., $\boldsymbol{\theta}^*$, on peut minimiser eq. (3.10) suivant la matrice de covariance. Le resultat obtenu est:

$$\mathbf{C}^* = \frac{1}{m}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))^\top \tag{3.12}$$

où $m = kn$ est le nombre total d'observations pour k images et n points 3-D.

Des formules similaires peuvent être obtenues dans le cas d'une matrice de covariance isotropique.

Maximum de vraisemblance avec des inliers et des outliers

Quand les outliers sont pris en compte, la méthode précédente ne peut pas être utilisée. Cependant, on peut utiliser l'estimation de la probabilité jointe des observations et de leurs valeurs. On obtient:

$$\begin{aligned}\log P_{\boldsymbol{\theta}}(\mathbf{S}, \mathbf{Z}) &= \sum_{i,j}\big(\delta_{in}(z_{ij})\log(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier})) \\ &+ \delta_{out}(z_{ij})\log(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{outlier})) + \text{const}\big)\end{aligned} \tag{3.13}$$

Cette expression ne peut pas être résolu comme précédemment, à cause de la présence des variables cachées z_{ij}. Pour résoudre le problème, on utilise le formalisme EM (maximisation de l'espérance), qui nous mène à:

$$Q_{EM} = \frac{1}{2}\sum_{i,j}\alpha_{ij}^{in}\Big((\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}) + \log(\det \mathbf{C})\Big) \tag{3.14}$$

3.2.2 Factorisation Affine avec l'algorithme EM

En combinant la factorisation affine et l'algorithme EM, $\min_{\boldsymbol{\mu}} Q_{EM}$ devient $\min_{\boldsymbol{\theta}} Q_{EM}$ et inclut $\mathbf{C}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i,j} \alpha_{ij}^{in} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{3.15}$$

En plus, la covariance qui minimise eq. (3.14) est:

$$\mathbf{C}^* = \frac{1}{\sum_{i,j} \alpha_{ij}^{in}} \sum_{i,j} \alpha_{ij}^{in} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))^\top \tag{3.16}$$

Des équations similaires peuvent être dérivées dans le cas où la covariance est isotrope.

Factorisation Affine avec EM:

Initialisation: Utilisez PowerFactorization [68] pour minimiser eq. (3.11), qui donne une estimation initiale pour $\boldsymbol{\theta}$ (les paramètres affines de la forme et du mouvement). Estimez $\mathbf{C}$ (matrice de covariance) en utilisant eq. (3.12).

Itère, jusqu'à convergence

Expectation: Mise à jour des valeurs de α_{ij}^{in} selon eq. (3.7) ou eq. (3.8).

Maximization: Trouvez $\boldsymbol{\theta}$ qui minimise Q_{EM} (factorization affine) selon eq. (3.15). Calculez la matrice de covariance $\mathbf{C}$ avec eq. (3.16). Alternativement, les équations pour covariance isotrope peuvent être utilisées.

Maximum a posteriori: Dès que EM converge, choisissez parmi un inlier ou un outlier pour chaque observation, i.e., $\max\{\alpha_{ij}^{in}; \alpha_{ij}^{out}\}$.

3.2.3 Factorisation Perspective Robuste

La méthode décrite dans la section précédente est incluse dans un algorithme [32, 173] qui obtient une reconstruction perspective et les paramètres extrinsèques des caméras à partir des caméras calibrées intrinsèquement. L'algorithme estime la différence entre le modèle perspectif faible et le modèle perspectif d'une manière itérative. Les détails de l'algorithme sont omis dans ce manuscrit.

3.2.4 Résultats

On a utilisé l'algorithme décrit dans deux contextes: l'étalonnage multi-caméras et la reconstruction 3-D éparse à partir de plusieurs caméras.

Étalonnage de plusieurs caméras

La figure 3.3 montre l'organisation partielle de la plate-forme multi-caméras ainsi que l'objet 1-D utilisé pour l'etalonnage. On utilise trois configurations de caméras, montrées en Figure 3.4: deux configurations de 30 caméras et une de 10 caméras. Nous les appelerons *Corner Case*, *Arc Case* et *Semi-Spherical Case*.

(a) (b)

Figure 3.3: (a): Vue partielle du montage de 30 caméras. (b): Les données pour l'étalonnage sont obtenues en déplaçant un objet unidimensionnel dans le champ visuel de toutes les caméras.

La figure 3.5 montre l'évolution de l'algorithme à partir de la solution perspective faible initiale vers la solution perspective. A la convergence, la solution trouvée par notre méthode (présentée en bleu-marin) est identique a la solution trouvée par l'ajustement de faisceaux (présentée en gris).

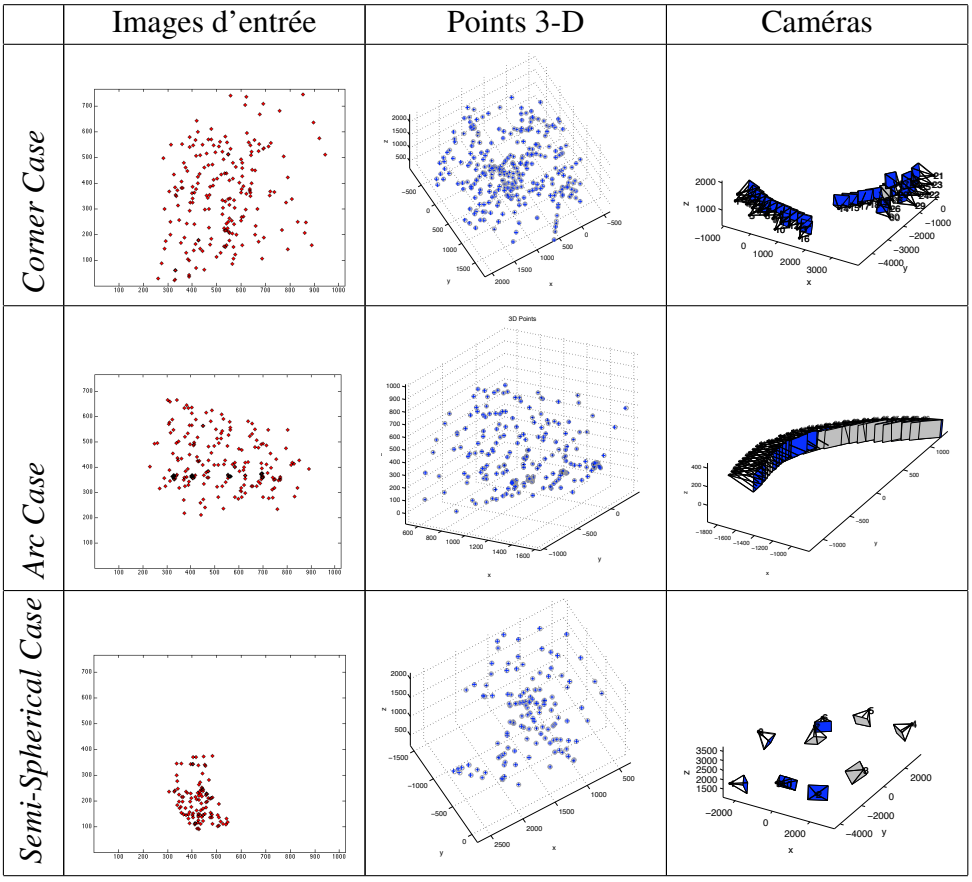


Figure 3.4: Résultats d'étalonnage multi-cameras. Gauche: Exemple typique de données d'entrée associées avec une caméra. Centre: Points 3-D reconstruits avec notre méthode (bleu-marin) et avec l'ajustement de faisceaux (gris). Droite: Résultats d'étalonnage obtenus avec notre méthode (bleu-marin) et avec l'ajustement de faisceaux (gris).

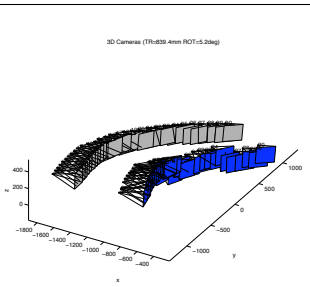
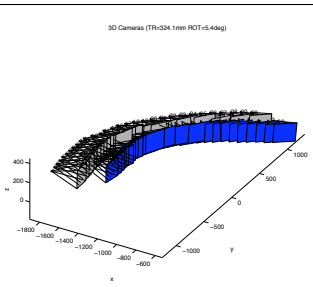
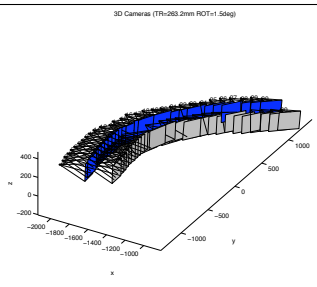
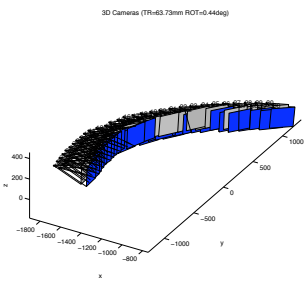
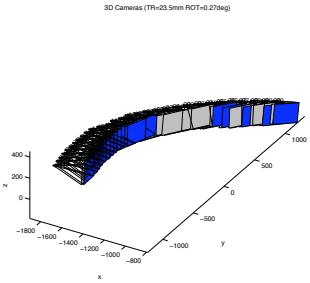
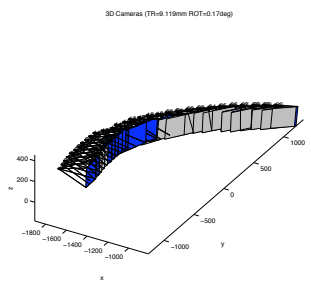
1^{ere} itération	2^{eme} itération	3^{eme} itération
		
2D Err. = 59.32	2D Err.= 18.43	2D Err. =15.92
4^{eme} itération	5^{eme} itération	8^{eme} itération
		
2D Err. = 4.53	2D Err. = 1.61	2D Err. = 0.24

Figure 3.5: Itérations de l'algorithme de factorisation perspective robuste dans le cas *Arc Case* et la comparaison avec l'ajustement de faisceaux . La première itération correspond au modèle perspectif faible. La solution pour l'ajustement de faisceaux est montrée en gris.

Etalonnage multi-caméras		*Corner Case*	*Arc Case*	*Semi-Spherical Case*
Données	# Cameras	30	30	10
	# Points 3-D	292	232	128
	# Prédictions 2-D	8760	6960	1280
	# observations manquantes	36%	0%	33%
	# Observations 2-D	5527	6960	863
Results	# Inliers 2-D	5202	6790	784
	# Inliers 3-D	285	232	122
	Erreur 2D (pixels)	0.30	0.19	0.48
	Erreur 3D (mm)	6.91	2.65	4.57
	Erreur rot. (dégrées)	0.13	0.18	0.27
	Erreur Tr. (mm)	27.02	9.37	24.21
	# iter. aff. (# iter. EM)	7 (2.4)	11 (2)	8 (3.2)

Table 3.1: Résumé de résultats obtenus pour l'étalonnage multi-cameras.

Etalonnage multi-caméras	*Corner Case*	*Arc Case*	*Semi-Spherical Case*
Méthode proposée	0.30	0.19	0.48
L'ajustement de faisceaux	0.58	0.61	0.95

Table 3.2: Comparaison entre la méthode proposée et l'ajustement de faisceaux pour les trois cas. L'erreur 2-D est mesurée en pixels.

Reconstruction 3-D

Nous avons testé la méthode proposée pour obtenir de reconstructions 3-D à partir de plusieurs caméras. On a utilisé plusieurs séquences, capturées avec une table tournante. En particulier, on a utilisé les jeux de données suivantes:

- "Dino" and "Temple" du site d'evaluation de Middlebury [1]
- "Oxford dinausor",[2]
- "Square Box" .

Nous avons utilisé l'implementation pyramidale OpenCV[3] du suiveur de points proposé par Lucas & Kanade [21], qui nous permet d'obtenir des matrices de mesures. La Figure 3.6 et le Tableau 3.3 résument des résultats obtenus.

[1] http://vision.middlebury.edu/mview/data/
[2] http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
[3] http://www.intel.com/technology/computing/opencv/

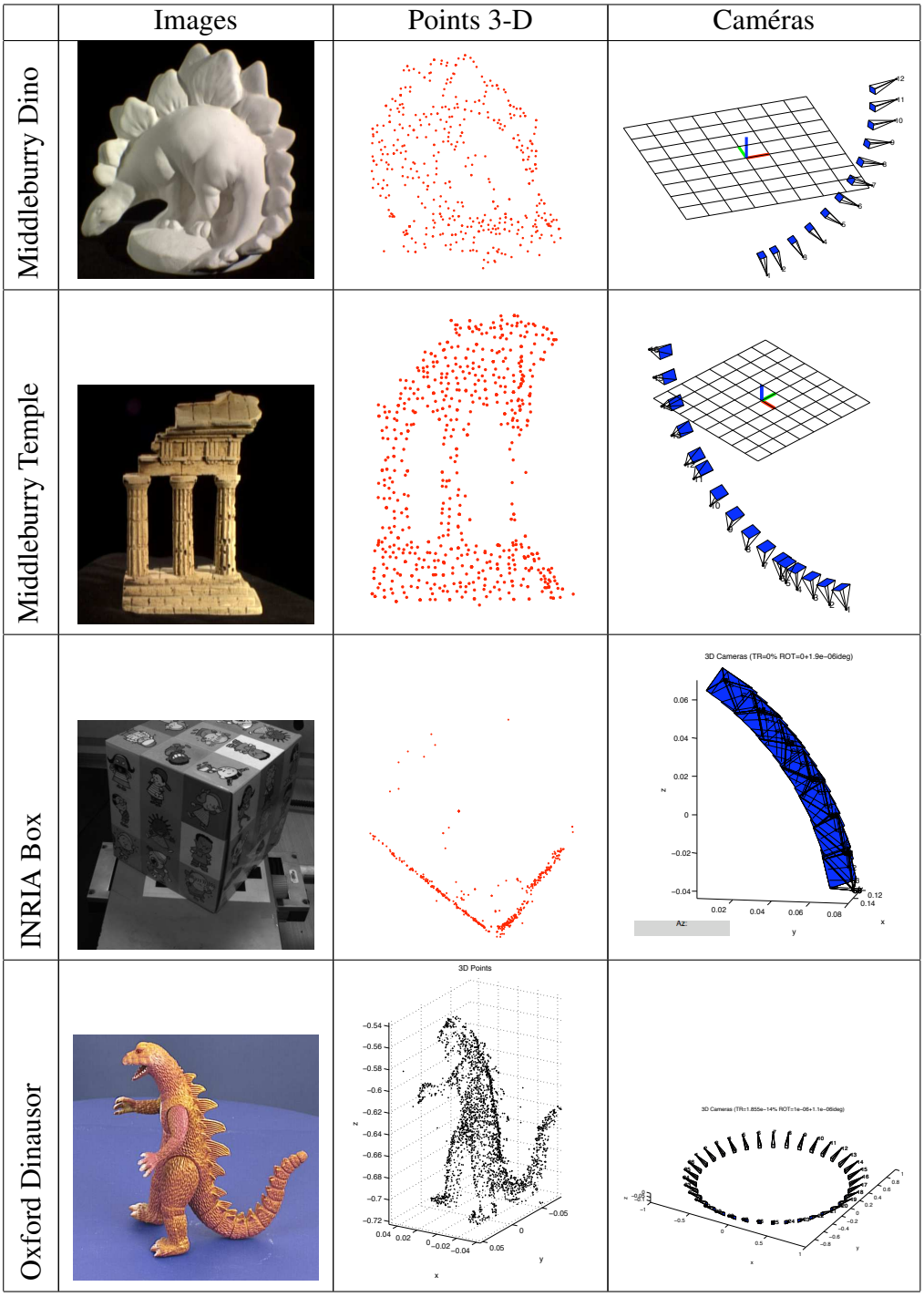


Figure 3.6: Les mesures réelles sont présentées en gris et les résultats de la reconstruction sont presentés en bleu marin.

3-D reconstruction		Dino	Temple	Box	Dinausor
Entrée	# Vues	12	16	64	36
	Taille de matrice **S**	24×480	32×758	128×560	72×1516
	# prédictions 2-D	5760	12128	35840	54576
	% d'observations manquantes	11%	21%	17%	85%
	# d'observations 2-D	5140	9573	29733	8331
Résultats	# inliers 2-D	3124	6811	25225	7645
	# inliers 3-D	370	720	542	1437
	Erreur 2D (pixels)	0.82	0.93	0.69	0.33
	Erreur rot. (degrees)	1.49	2.32	–	0.00
	Erreur trans. (mm)	0.01	0.01	–	0.00
	# iter. aff.(# iter. EM)	7 (4)	7 (3.14)	9 (3)	7 (3.29)

Table 3.3: Résumé de la reconstruction.

3.3 Méthode d'évolution de maillages

Dans cette partie, une méthode d'évolution de maillages est proposée. Elle est capable de gérer les changements topologiques et les auto-intersections sans imposer de contraintes d'échantillonnage sur le maillage. La géométrie exacte du maillage est préservée, à l'exception de parties qui s'auto-intersectent; elles sont retriangularisées localement. Des applications sont présentées: le morphing des maillages et la reconstruction 3-D à partir de plusieurs caméras en utilisant des méthodes variationnelles.

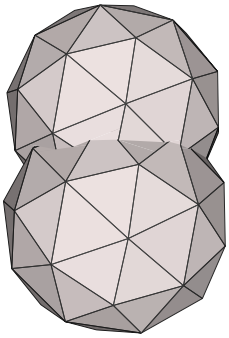
3.3.1 Méthode

Le principe de la méthode, intitulé TransforMesh, est de trouver un triangle racine initial (seed-triangle), situé à l'extérieur, qui ne s'entrecoupe pas avec des autres triangles, pour ensuite propager cette information de l'extérieur au moyen de croissance de région. On appelle un triangle *valide* un triangle qui est situé a l'exterieur et qui ne s'entrecoupe pas avec les autres triangles. Un triangle *partiel* est un triangle qui s'entrecoupe avec les autres, mais qui a une partie à l'extérieur. L'algorithme, résumé en Figure 3.7, utilise 3 listes pour cet objectif: une qui s'appelle $\mathcal{V}$ de triangles valides; une qui s'appelle $\mathcal{P}$ de triangles partiels et finalement une qui s'appelle $\mathcal{G}$ où tous les triangles et sous-triangles valides seront stockés avant d'être finalement collés dans un nouveau maillage. Les triangles partiels sont retriangularisés en utilisant une triangulation Delaunay 2-D avec des contraintes, où les contraintes sont tous les segments des intersections avec les autres triangles. Un exemple est illustré dans la Figure 3.8.

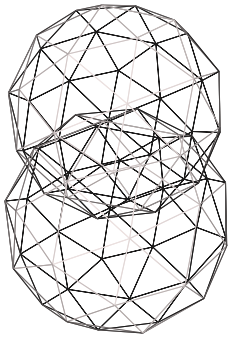
TransforMesh - Entrée: maillage triang. $\mathcal{I}$; Sortie: maillage triang. $\mathcal{O}$

1. Calcul des intersections - calculez toutes les intersections entre tous les triangles qui font partie du maillage $\mathcal{I}$
2. Croissance d'une région valide
   - 2.1. Initialisation - marquez tous les triangles non visités
   - 2.2. Découverte d'un triangle racine initial - trouvez un triangle valide extérieur en $\mathcal{I}$ et ajoutez à $\mathcal{V}$
   - 2.3 Tant que $\mathcal{V} \neq \emptyset$
     - a. Tant que $\mathcal{V} \neq \emptyset$ ou $\mathcal{P} \neq \emptyset$
       - a.1. Traitement liste valide - $\forall t \in \mathcal{V}$, ajoutez t à $\mathcal{G}$; ajoutez les voisins valides non-visités de t à $\mathcal{V}$ et les voisins partiels à $\mathcal{P}$ en gardant l'arrêt d'entrée.
       - a.2. Traitement liste partielle - $\forall t \in \mathcal{P}$, obtenez une triangulation Delaunay avec contraintes (intersection segments + arrêtes); commencez avec l'arrêt d'entrée et sélectionnez des triangles en les ajoutant à $\mathcal{G}$; arrêtez vous sur les arrêtes avec des contraintes; ajoutez les triangles de l'autre côté des arrêtes avec contraintes à la liste appropriée, si pas encore visités.
     - b. Découverte d'un triangle racine initial - trouvez un triangle valide extérieur en $\mathcal{I}$ et ajoutez le à $\mathcal{V}$
   - 2.4 Collage de triangles - on calcule le maillage $\mathcal{O}$ à partir de $\mathcal{G}$

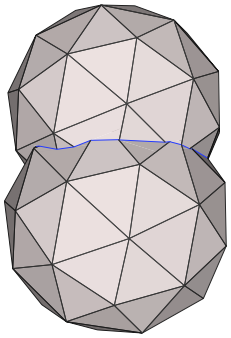
Figure 3.7: Résumé de l'algorithme. En plus, la Figure 3.8 montre les étapes principales de la méthode.



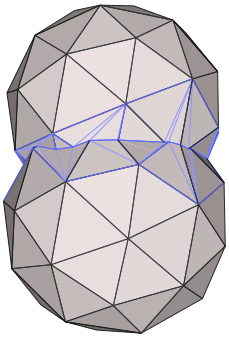
(a) Entrée (ombragé)



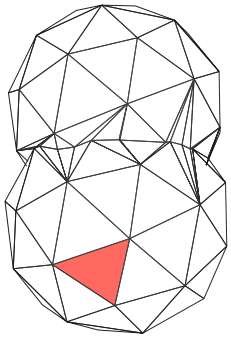
(b) Entrée (filaire)



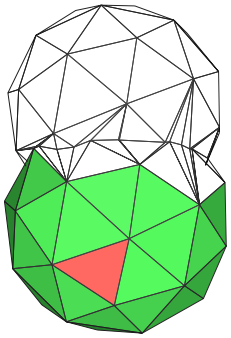
(c) Intersections des facettes



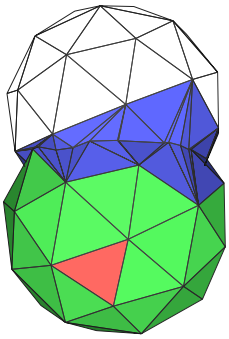
(d) Segments Delaunay 2-D



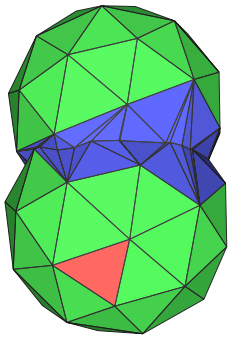
(e) Découverte d'un triangle racine initial



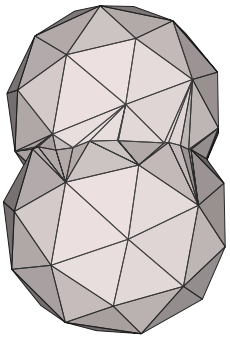
(f) Expansion de triangles valides



(g) Expansion de triangles partielles



(h) Fin d'expansion



(i) Sortie(ombragé)

Figure 3.8: Un example de TransforMesh, l'algorithme proposé.

3.3.2 Evolution de maillage générique

TransforMesh peut être utilisé de manière générique dans les problèmes qui nécessitent des évolutions de surfaces. L'algorithme est présenté dans Figure 3.9.

Évolution de maillages générique en utilisant TransforMesh

Tant que non terminé

1. **Calcul du Champ de Velocité** $\vec{\mathcal{F}}$ pour chaque sommet du maillage $\mathcal{M}$, en utilisant l'information spécifique au domaine d'application.
2. **Evolution de maillage** $\mathcal{M}$ en utilisant le champ de vecteurs $\vec{\mathcal{F}}$ et un pas de temps t, avec un seuil maximal $\alpha \cdot e_{avg}(v)$, où α est un seuil fixé par l'utilisateur (typiquement dans l'intervale 0.1-0.3) et $e_{avg}(v)$ est la taille moyenne locale de sommet v.
3. **Invoquez TransforMesh** sur $\mathcal{M}$ pour enlever les problèmes liés au changement de topologie est auto-intersections.
4. **Optimisation du maillage**
   a) *Re-maillage Adaptatif*: garantit que toutes les arrêtes sont dans un intervalle de sûreté, i.e. $\forall e \in \mathcal{M}, e \in [e_1, e_2]$, en faisant les opérations d'échange des arrêtes et effondrement des arrêtes.
   b) *Optimisation de la valence d'un sommet*: en faisant les opérations d'échange des arrêtes pour qu'il obtienne une valence de 6 pour chaque sommet [87].
   c) *Lissage Laplacien du maillage*: chaque sommet v est mis-a-jour par $v \rightarrow v - \beta\Delta v$, où Δv est la representation discrète du Laplacien maillage [36, 113].

Figure 3.9: Algorithme d'evolution maillage generique en utilisant TransforMesh.

3.3.3 Résultats

Comme applications, on va utiliser l'algorithme générique dans les contextes de morphing du maillage et de reconstruction 3-D dense à partir de plusieurs

caméras.

Morphing du maillage

Dans le morphing du maillage, on commence avec une surface initiale S_A qui évolue vers une surface cible S_B. En utilisant le morphing du maillage, on peut tester différents cas de topologies complexes. Dans cette implémentation, on a adopté la méthode proposée par Breen et al. [24].

Quelques résultats sont montrés dans la Figure 3.4, 3.6 et le Tableau 3.5.

Réconstruction 3-D dense

La motivation originale pour développer un algorithme qui permet d'enlever les auto-intersections et qui gère les changements de topologie était de récupérer une surface 3-D à partir de plusieurs caméras en utilisant des méthodes d'évolution variationelles.

Dans notre cas, on commence l'évolution à partir de l'enveloppe visuelle, obtenue avec la méthode décrite en [49], qui utilise l'information de silhouette. La surface de départ est déformée pour améliorer une fonctionnelle prenant en compte la photo-consistance. Cette mesure photométrique est détaillé en [131], où les auteurs utilisent une formulation level-set. Notre but était de remplacer la méthode implicite utilisée en [131] en gardant la même fonctionnelle d'énergie.

On a testé l'algorithme de reconstruction 3-D sur les données du site Middlebury [4] [142] et on a établi le fait que nos résultats sont comparables avec l'état de l'art, en obtenant une précision sous-millimetrique. Quelques résultats sont aussi presentés dans le Tableau 3.7. On a aussi inclut les résultats obtenus par Furukawa et al. [54], Pons et al. [131] et Hernandez et al. [70], considérés comme l'état de l'art. Les differences sont très faibles, entre $0.01mm$ et $0.3mm$. Des résultats de reconstruction sont montrés dans la Figure 3.10.

La séquence *Leuven* [5] contient 7 images de haute résolution 3000×2000 de la Préfecture de Leuven. La reconstruction finale contient un maillage avec plus de 1.5 million de triangles. Les résultats sont presentés dans la Figure 3.11. Une procédure de re-maillage adaptative a été utilisée pour que les triangles invisibles ne soient pas réduits.

[4] http://vision.middlebury.edu/mview/
[5] http://cvlab.epfl.ch/data/strechamvs/

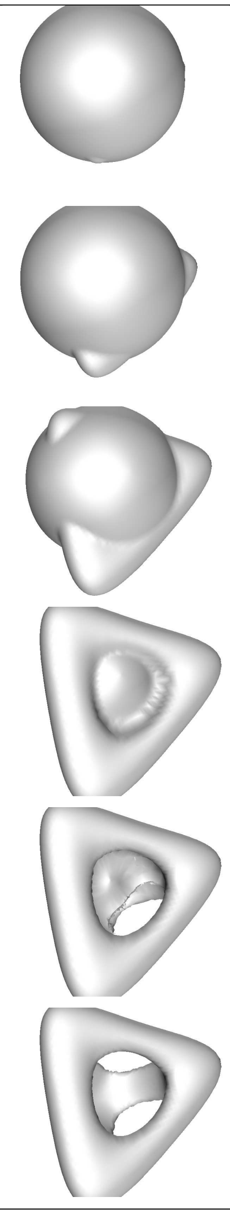
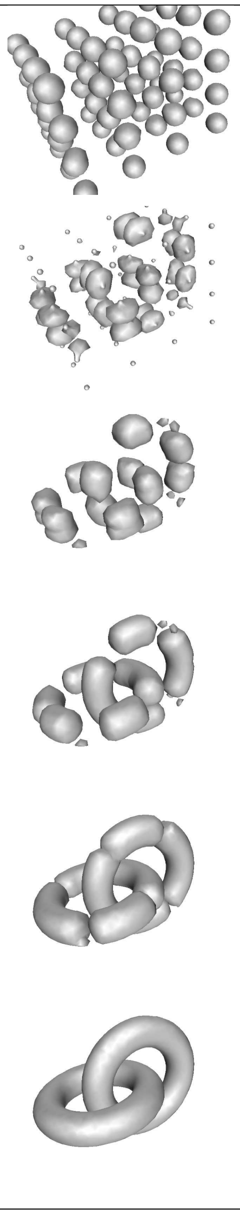
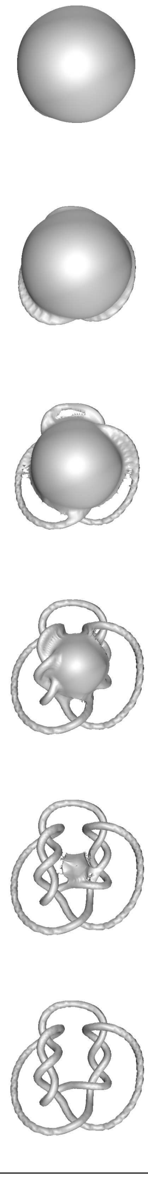
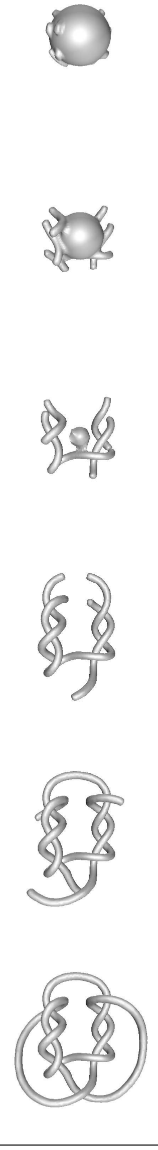
Genus 3	Toruses	Knots In	Knots Out
			

Table 3.4: Exemples de morphing du maillage avec des surfacess fermées.

Donées	Iterations	# Facettes	# Intersections	Temps (intersect.)	Temps (total)
Genus 3	54	4764.14	33.88	0.65 sec	1.42 sec
Toruses	37	6296.33	22.67	0.81 sec	1.78 sec
Knots In	119	13244.25	101.52	1.63 sec	3.58 sec
Knots Out	430	3873.11	4.86	0.18 sec	0.89 sec

Table 3.5: Statistiques pour les différents cas presentés dans le Tableau 3.4 Le temps de calcul correspondent à une machine avec un processeur Intel Core2Duo 2.6 GHz. Les 4 dernièrs colonnes représentent des mesures moyennes.

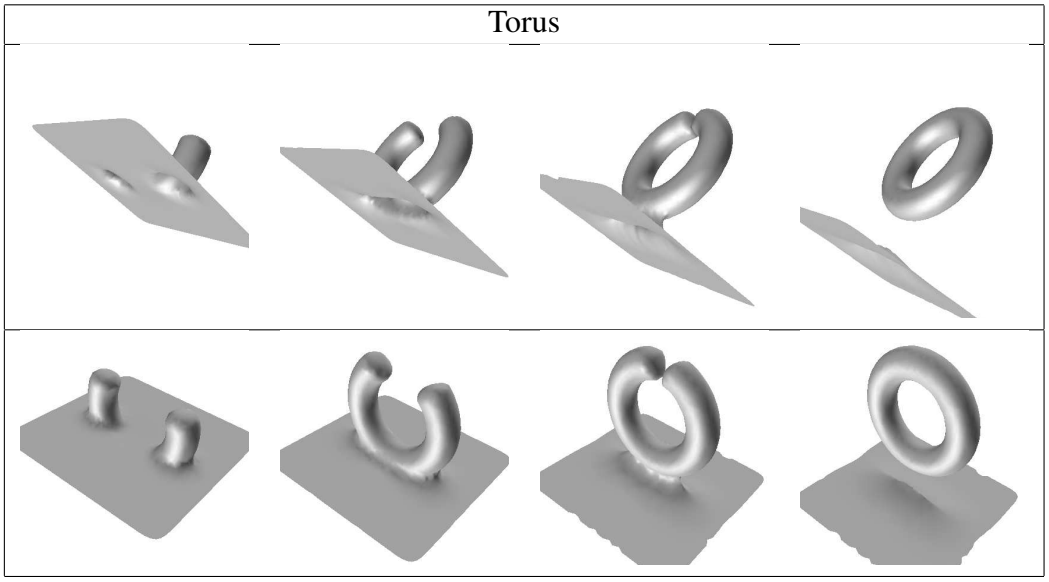
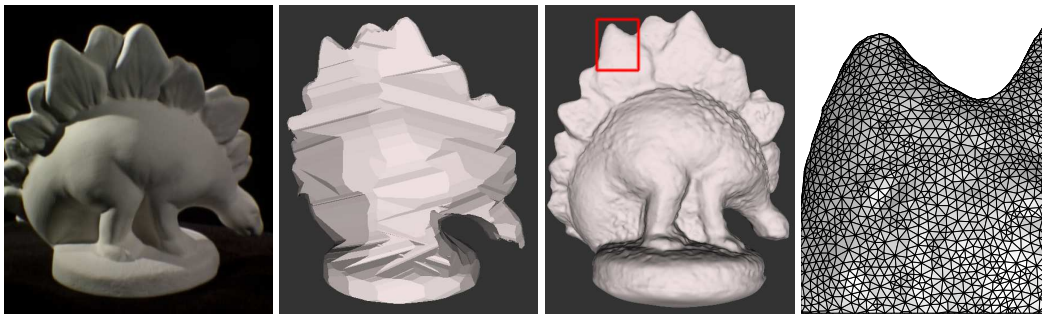


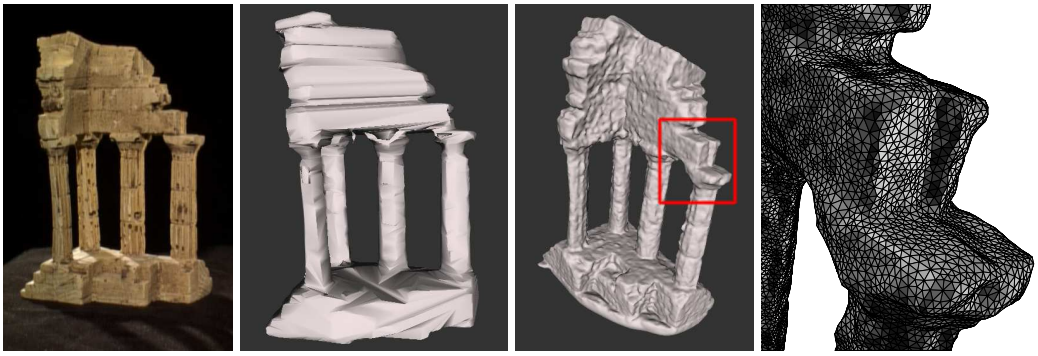
Table 3.6: Résultat du morphing du maillage avec une surface ouverte.

Données / Méthode	Temple Ring		Temple Sparse Ring		Dino Ring		Dino Sparse Ring	
	Prec.	Compl.	Prec.	Compl.	Prec.	Compl.	Prec.	Compl.
Pons et al. [131]	0.60mm	**99.5%**	0.90mm	95.4%	0.55mm	99.0%	0.71mm	97.7%
Furukawa et al. [54]	0.55mm	99.1%	**0.62mm**	**99.2%**	**0.33mm**	**99.6%**	**0.42mm**	**99.2%**
Hernandez et al. [70]	**0.52mm**	**99.5%**	0.75mm	95.3%	0.45mm	97.9%	0.60mm	98.52%
Nos résultats	0.55mm	99.2%	0.78mm	95.8%	0.42mm	98.6%	0.45mm	**99.2%**

Table 3.7: Résultat de la reconstruction 3-D. **Précision**: la distance d en mm qui amène 90% de résultat R proche de la vraie surface G. **Complétude**: le pourcentage de G qui se trouve à moins de 1.25mm de R.



(a) Dino - image (b) Dino - debut (c) Dino - final (d) Dino - de près



(e) Temple - image (f) Temple - debut (g) Temple - final (h) Temple - de près

Figure 3.10: Résultats obtenus dans les cas du *temple* et du *dino*.

La séquence *Staty* contient 32 images de résolution 720×576. Les résultats de reconstruction sont présentés dans la Figure 3.12. On montre également la courbure moyenne pour souligner les parties de la reconstruction avec des structures assez fines, comme le pavé.

Pour faire le lien avec la section précèdente, on a utilisé les reconstructions éparses de points 3-D obtenues avec la méthode de factorisation perspective robuste présentée dans la Section 3.2. On l'a combiné avec PowerCrust [12], un algorithme qui génère un maillage à partir de point 3-D, pour obtenir une surface de départ pour l'algorithme de reconstruction 3-D qui prenne en compte la photo-consistance. Les résultats sont montrés dans la Figure 3.13.

3.4 Regroupement de caméras

Une méthode de regroupement de caméras qui utilise l'information de la scène est développée, capable de séparer des reconstructions à grande échelle, qui consomment beaucoup de mémoire, en plusieurs petites tâches indépendantes de reconstructions partielles utilisant moins de ressources.

3.4.1 Méthode

L'idée principale est de générer une matrice de visibilité Δ obtenue à partir des points d'une reconstruction éparse pour ensuite créer des regroupements de points en prenant compte les lignes ou les colonnes de la matrice de visibilité. Chaque regroupement de caméras est traité ultérieurement d'une manière indépendante. La méthode proposée peut être utilisée avec n'importe quelle méthode de reconstruction 3-D à partir de plusieurs caméras étalonnées. Dans notre cas, on a utilisé la méthode proposée dans la Section 3.3. Un résumé de la méthode est présentée dans la Figure 3.14.

Regroupement de caméras à partir des caméras

Si on utilise les colonnes de la matrice Δ quand on calcule le regroupement, on obtient directement un regroupement de caméras. Des exemples sont montrés dans la Figure 3.15.

Regroupement de caméras a partir de la géométrie

Si on utilise les lignes de la matrice Δ, on obtient des regroupements de sommets de la géométrie 3-D, il faut donc choisir les caméras avec le plus de votes pour chaque regroupement. Pour le dernier pas, un seuil α doit être choisi pour distinguer les caméras les plus discriminatoires. Cette dernière méthode nous permet d'avoir des chevauchements sur les regroupements de caméras. Des exemples sont montrés dans la Figure 3.16.

3.4.2 Résultats

Séquence Dino et Temple. On a utilisé les séquences *dino* et *temple* du site Middleburry [142], qui ont 47 images de taille 640x480. Les résultats pour 2 regroupements sont montrés dans la Figure 3.17 et au Tableau 3.8 (regarder [6] pour plus d'information). Comme on peut l'observer dans le Tableau 3.8, notre méthode ne perd pas en précision d'un façon significatif en utilisant le regroupement de caméras, de plus elle gagne visiblement au niveau de la consommation mémoire.

Données / Méthode	Temple Ring				Dino Ring			
	Prec.	Compl.	Mem.	Durée	Prec.	Compl.	Mem.	Durée
Pas de regroupement	0.55mm	99.2%	1031MB	60min	0.42mm	98.6%	962MB	43min
Regroupement 1/2	0.62mm	98.5%	468MB	36 min	0.5mm	98.5%	483MB	33min
Regroupement 2/2			472MB	42 min			476MB	35min

Table 3.8: Résultats pour les données de Middleburry. **Précision**: la distance d en mm qui amène 90% de résultat R proche de la vrai surface G. **Complétude**: le pourcentage de G qui se trouve a moins de 1.25mm de R. **Memoire**: le quantité de mémoire RAM utilisée par le logiciel. **Durée**: le temps pris par le logiciel pour finir.

On montre aussi des résultats pour deux autres séquences: *parthenon* et *fontaine*. La séquence *parthenon* utilise 200 cameras de résolution 640×480. Les résultats pour 21 regroupements sont présentés dans la Figure 3.18. En plus, on considère le cas où au lieu d'avoir beaucoup de caméras avec une résolution moyenne, on a une quantité moyenne de caméras à haute resolution. Dans la séquence *fontaine* il y a 11 caméras à 3072×2048. Dans la reconstruction par sous-parties, on a considéré 4 regroupement. On a utilisé le regroupement sur la géométrie et on a généré des caméras virtuelles en prenant compte la boite délimitante de tous les points de chaque encadrement projeté. Les résultats sont montrés dans la Figure 3.19.

[6] http://vision.middlebury.edu/mview/eval/

3.5 Descripteur 3-D sur les maillages

Dans cette section, un nouveau descripteur en 3 dimensions est proposé, défini sur des maillages triangulaires échantillonnés de façon uniforme. Ce descripteur est invariant en rotation, translation et échelle et permet de capturer les informations géométriques et photométriques locales. Il est particulièrement utile dans un environnement multi-caméra, où les maillages reconstruits sont dotés avec la couleur / texture. De plus, le descripteur est défini d'une manière générique pour une fonction quelconque, définie sur la surface (i.e. la couleur, la courbure). Des résultats de correspondance rigide et non rigide sont présentés. Finalement, le descripteur est intégré dans un cadre de suivi temporel dense du maillage.

3.5.1 Méthode

On considère un maillage triangulaire échantillonné de manière uniforme et une fonction scalaire $f(v)$ quelconque définie sur les sommets v de la surface S. La fonction peut représenter differents type d'information, comme la couleur (dans un contexte multi-caméras) ou la courbure moyenne.

On introduit l'opération de convolution (avec un noyau gaussien) de la fonction f, defini sur le maillage. Le gradient local directionnel de la fonction et le gradient local moyen de la la fonction f dans le voisinage d'un sommet v sont aussi définis.

Détecteur de points d'intérêt

On adopte et on adapte le formalisme d'espace d'échelle sur les maillages réguliers en utilisant l'opération de convolution. Les points d'intérêt considérés sont les extrémités de l'espace d'échelle. Un seuil est mis en place pour garder les 5% extrémités les plus importantes . Ultérieurement, les extrémités sont filtré par un détecteur de coins, en calculant la matrice hessienne des dérivées partielles secondaires, pour garder les points d'intérêt les plus stables. Un exemple est présenté dans la Figure 3.20.

Le Descripteur

Le descripteur est une adaptation sur les maillages des descripteurs HOG (Histogram of Oriented Gradient) [33] ou SIFT(Shift Invariant Feature Transform) [97], définis sur des images.

Le descripteur proposé, nommé MeshHOG, est défini sur le maillage pour un sommet v et un voisinage contenant r_n anneaux autour de v. Comme dans le cas de HOG, l'idée principale est de calculer des histogrammes des gradients locaux en gardant un niveau de support spatial. Comme dans notre scénario les gradients locaux sont des vecteurs 3-D, les histogrammes sont aussi définis en 3-D.

Mise en correspondance

On considère deux surfaces a mettre en correspondance pour lesquelles on a calculé les points d'intérêt et les descripteurs associés. Pour chaque descripteur de la première surface on calcule la distance de tous les descripteurs de l'autre surface, en gardant les deux meilleurs résultats. On considère un seuil minimal sur le rapport entre les deux meilleures distances, pour imposer une discrimination minimale.

3.5.2 Résultats

Quelques résultats sont présentés dans la Figure 3.22, où des maillages rigides, obtenus par des reconstructions 3-D, sont mis en correspondance. Différentes mesures de qualité, illustrées dans la Figure 3.21, peuvent être utilisées pour la détection de points d'intérêt et pour le descripteur.

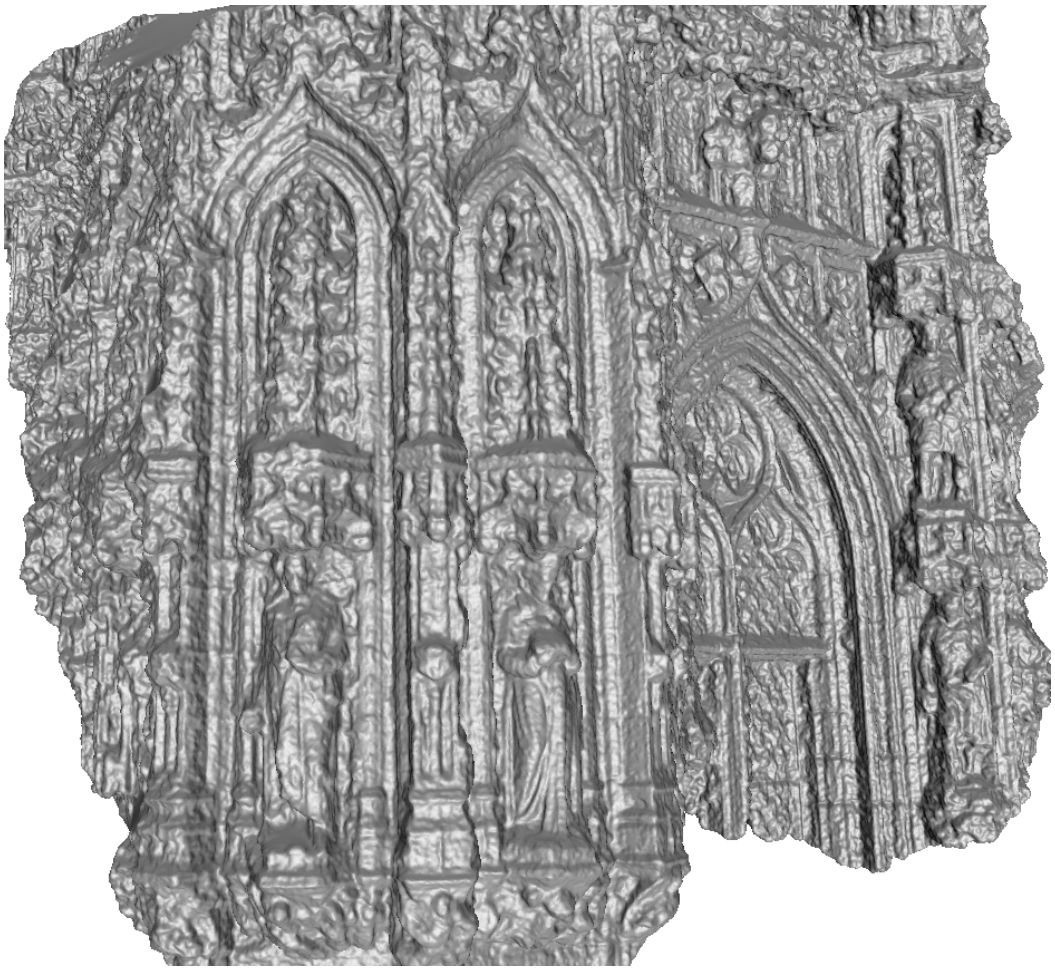
Intégration avec une procédure de suivi de maillage dense

On a aussi integré le descripteur MeshHOG avec une procédure de suivi de maillage dense, qui a été décrit dans [160]. Quelques résultats sont présentés dans la Figure 3.23.

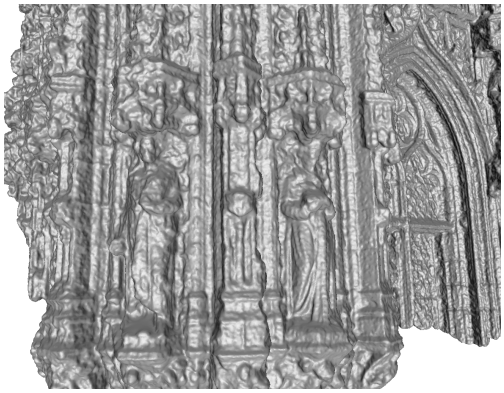
3.6 Conclusion

Dans cette thèse nous avons proposé de nouvelles solutions pour chacun des composants d'un système de reconstruction spatiale et temporelle à partir des plusieurs caméras: l'étalonnage de caméras et reconstruction spatiale éparse, la reconstruction spatiale dense, la reconstruction temporelle éparse et la reconstruction temporelle dense.

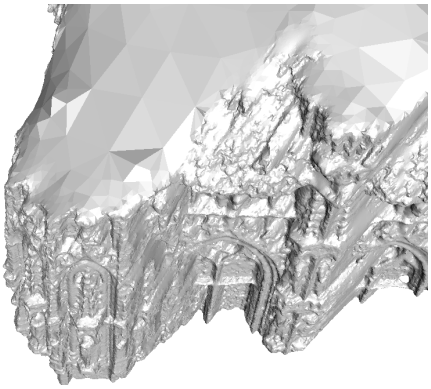
(a) Images d'entrée (3000×2000)



(b) Vue d'ensemble

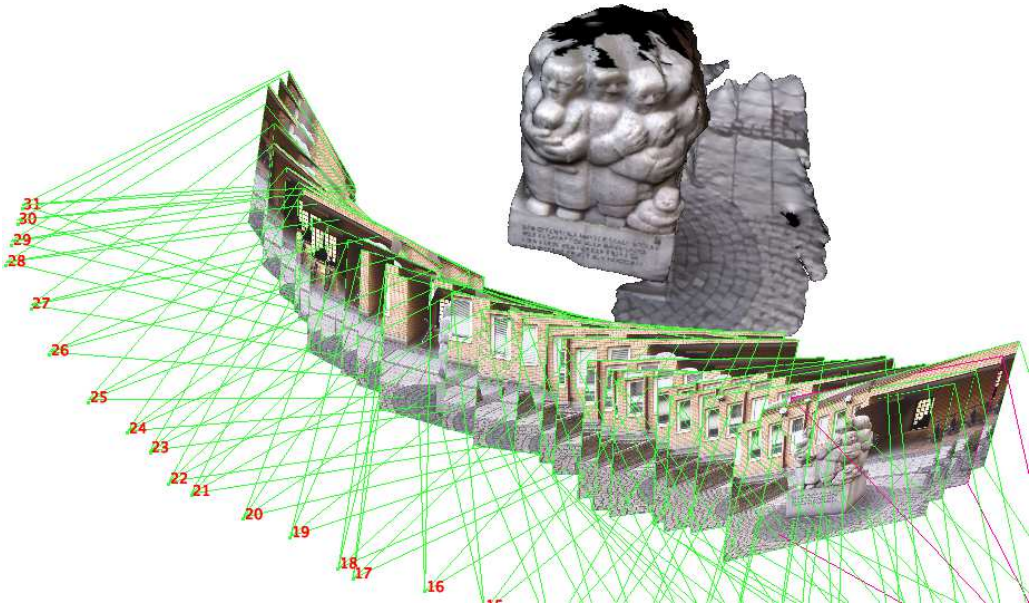


(c) Vue de près



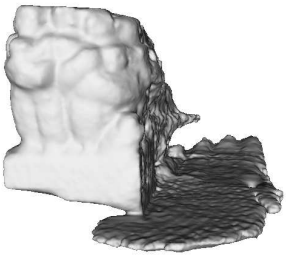
(d) Vue d'en haut

Figure 3.11: Résultats obtenus pour la séquence *Leuven*. Les triangles invisibles sont maintenus fixes, comme on peut observer en (d), ce qui fait que la représentation reste la plus compacte possible.

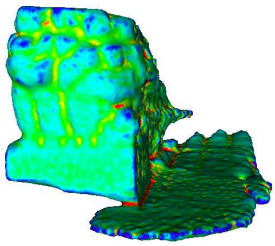


(a) Vue d'ensemble

(b) Vue 1 - texturé

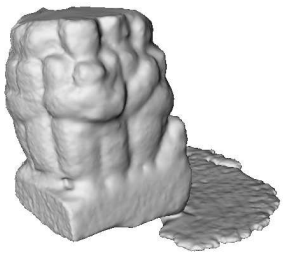


(c) Vue 1 - blanc

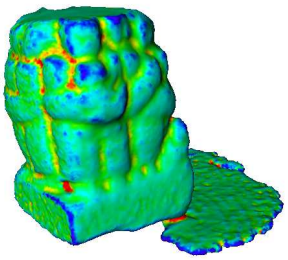


(d) Vue 1 - courbure

(e) Vue 2 - texturé



(f) Vue 2 - blanc



(g) Vue 2 - courbure

Figure 3.12: Résultats pour la séquence Staty.

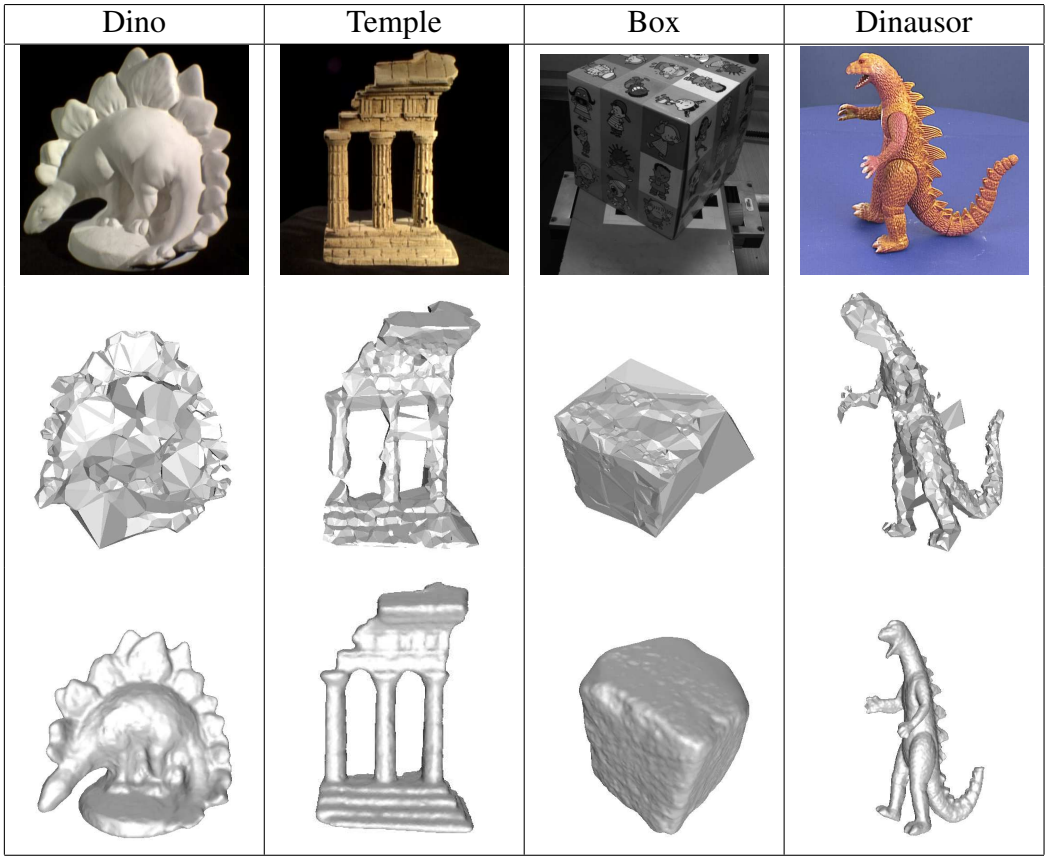


Figure 3.13: Résultats de reconstructions dénse montrant le lien avec la section précédente. Ligne en haut: Exemples des images d'entrée; Ligne au milieu: un maillage brut obtenu en utilisant PowerCrust [12] avec des points éparses 3-D qui sont la solution de l'algorithme de factorisation perspective robuste proposé dans la section précédante. Ligne en bas: reconstructions finales obtenus avec la méthode proposés dans cette section.

Regroupement de caméras

1 On détermine **une matrice de visibilité** Δ de points :

   a) en obtenant une reconstruction 3-D éparse et en projetant tous les sommets dans les images en prenant compte de la visibilité

   b) en trouvant des points d'intérêt et en les mettant en correspondance au moyen d'un descripteur (i.e. SIFT [97]), comme il a déjà été proposé en [146]

$$\Delta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,N_c} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,N_c} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N_v,1} & \beta_{N_v,2} & \cdots & \beta_{N_v,N_c} \end{bmatrix}$$

   où N_v est le nombre de sommets et N_c est le nombre de caméras.

2 On obtient des **regroupements de caméras en faisant un regroupement k-means** [17] sur les lignes ou les colonnes de matrice Δ.

3 **Pour chaque regroupement de caméras, on fait une reconstruction 3-D à haute résolution** en gardant les sommets invisibles fixes et en minimisant seulement les parties visible pour ce regroupement.

Figure 3.14: Méthode de regroupement de caméras

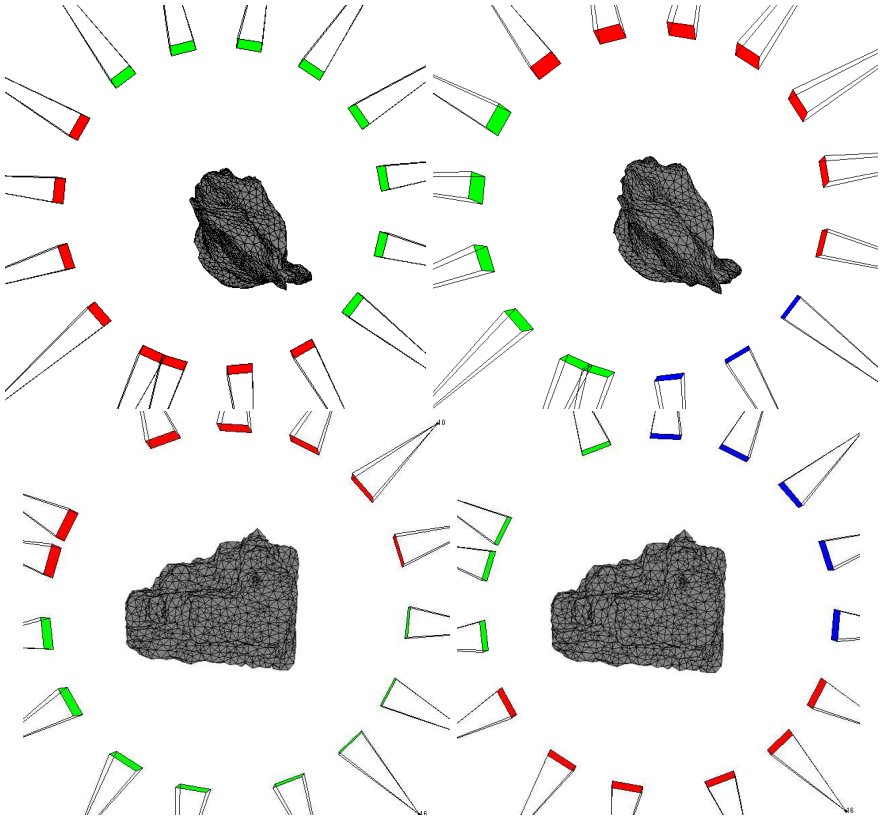


Figure 3.15: Example de *regroupement de caméras à partir des caméras* en utilisant deux ensembles de données, dino et temple, avec 2 et 3 regroupements.

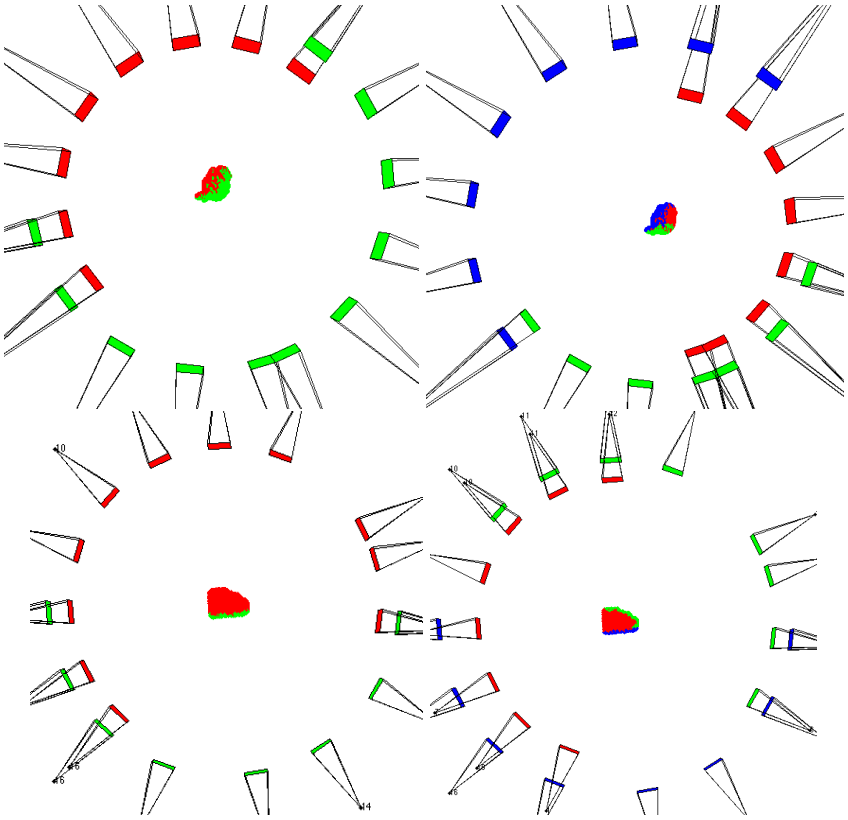


Figure 3.16: Example de *regroupement de caméras à partir de la géométrie* en utilisant deux ensembles de données, dino et temple, avec 2 et 3 regroupements.

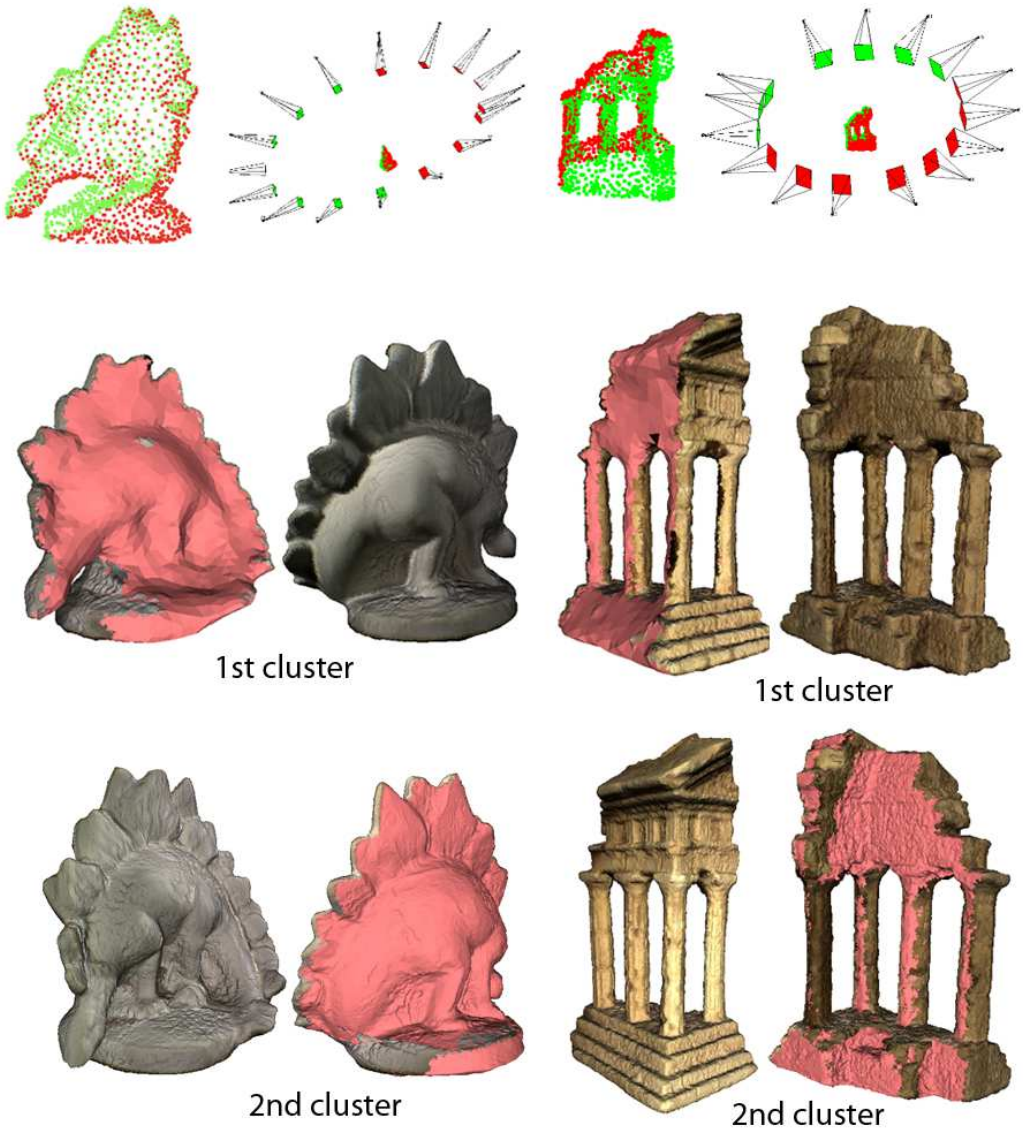


Figure 3.17: Exemple de cas Dino et Temple. La ligne en haut: regroupement des caméras; la ligne au milieux et la ligne en bas: des reconstructions partielles. Les sommets invisibles pour chaque regroupement sont présentés en rouge pâle.

Figure 3.18: Résultats avec 21 regroupements de la reconstruction de la séquence du parthenon avec 200 caméras de résolution 640×480.

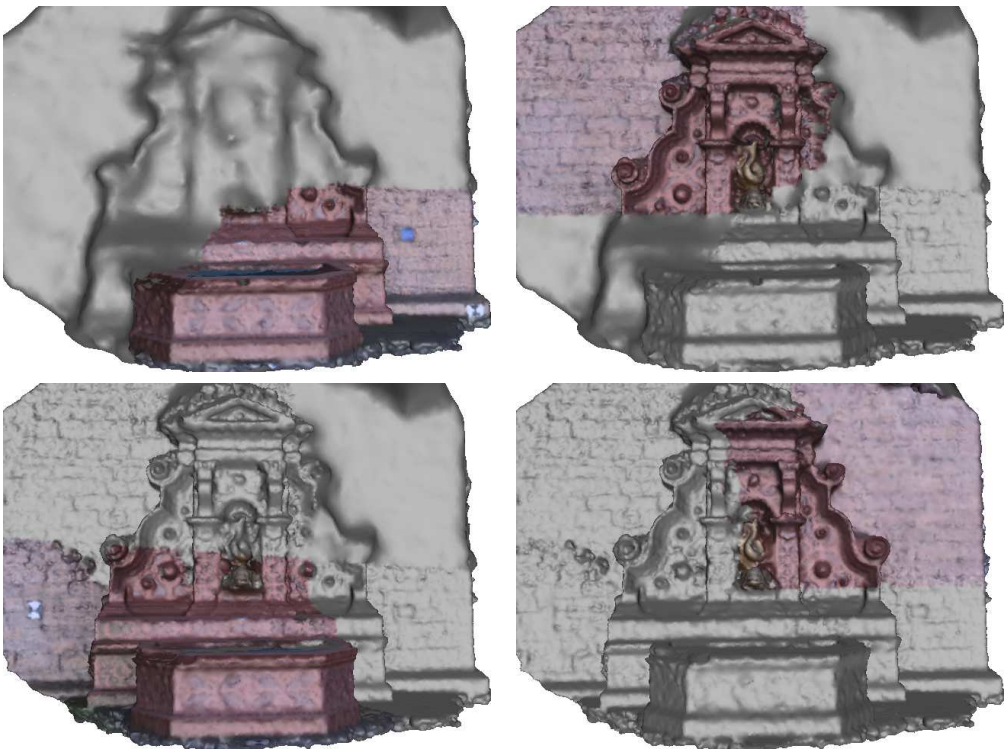
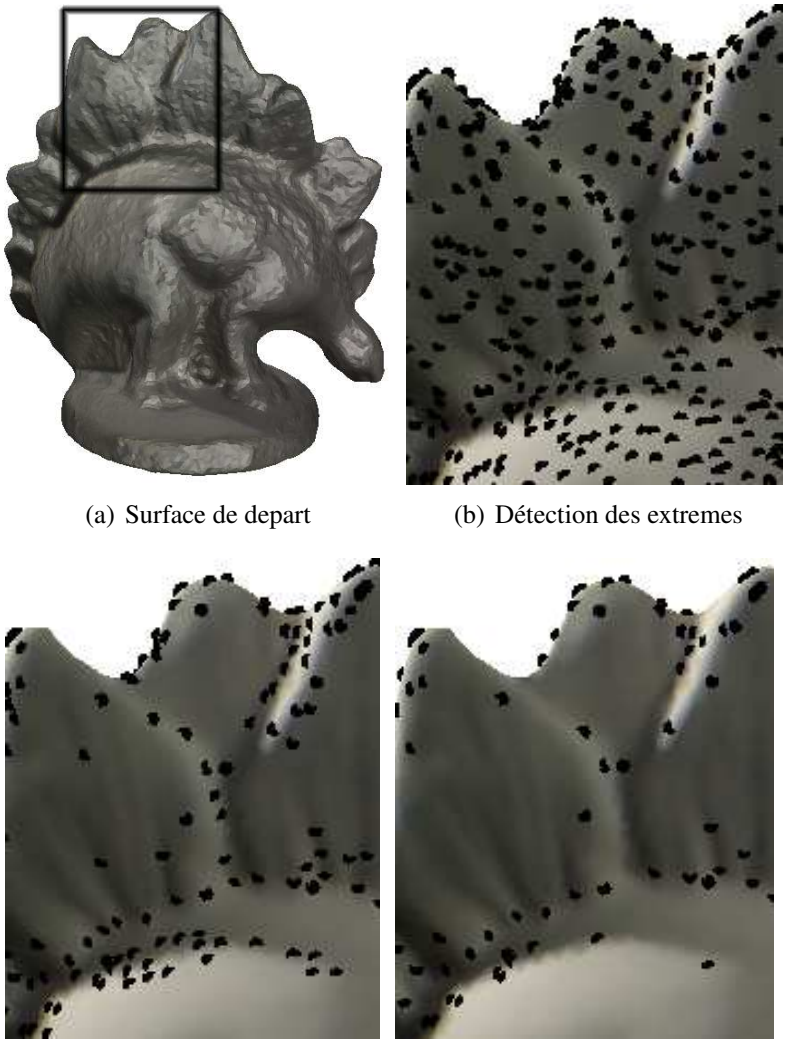


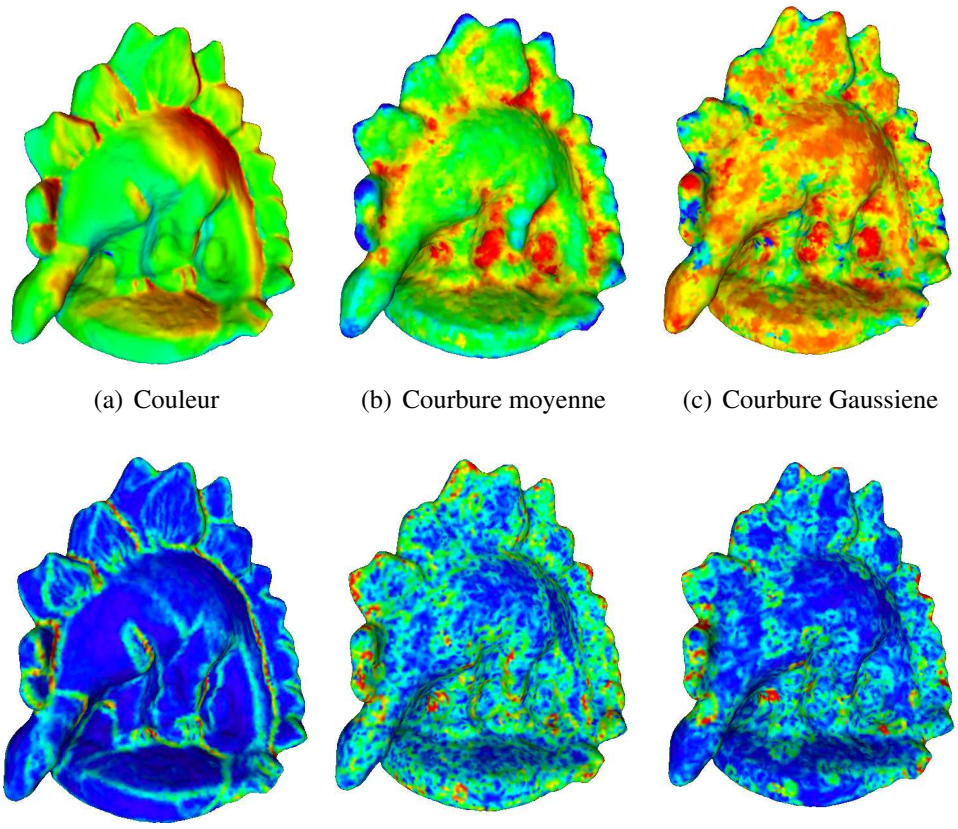
Figure 3.19: Résultats avec 4 regroupements de la reconstruction de la séquence de la fontaine avec 11 caméras de résolution 3072×2048. Des caméras virtuelles sont générées pour chaque regroupement en projetant les points de la reconstruction éparse et prenant en compte de la boite délimitante.



(a) Surface de depart (b) Détection des extremes

(c) 5% les plus grandes extrêmes (d) Coins

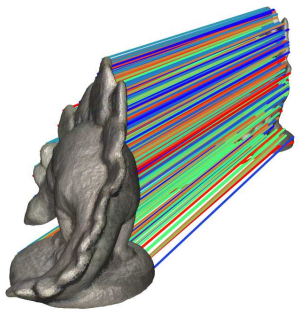
Figure 3.20: Les étapes pour trouver les points d'intérêt. La qualité utilisée est la *couleur*. Le maillage original a 27240 sommets. Le détecteur des extrémités trouve 5760 points. En mettant en place un seuil qui garde les 5% plus importants parmi les sommets, il reste1360 sommets. Après considération des coins, il reste 650 sommets.



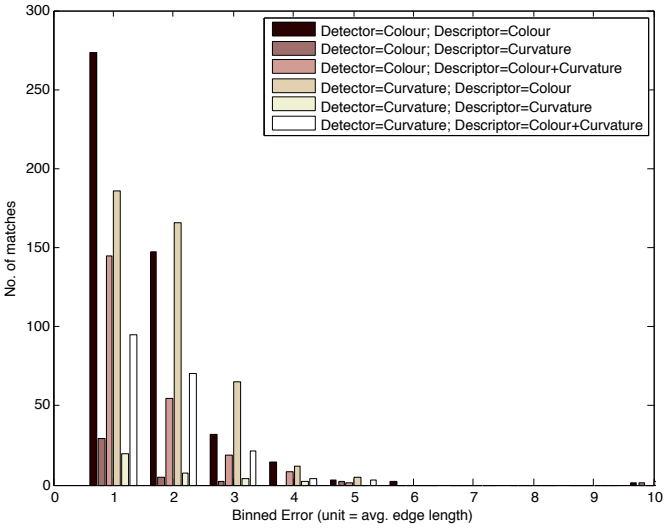
(a) Couleur (b) Courbure moyenne (c) Courbure Gaussiene

(d) Gradient de la couleur (e) Gradient de la courbure moyenne (f) Gradient de la courbure Gaussienne

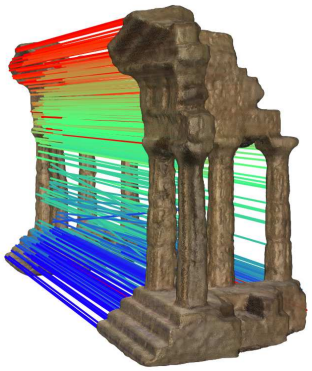
Figure 3.21: Différentes mesures de qualité pour le *Dino*.



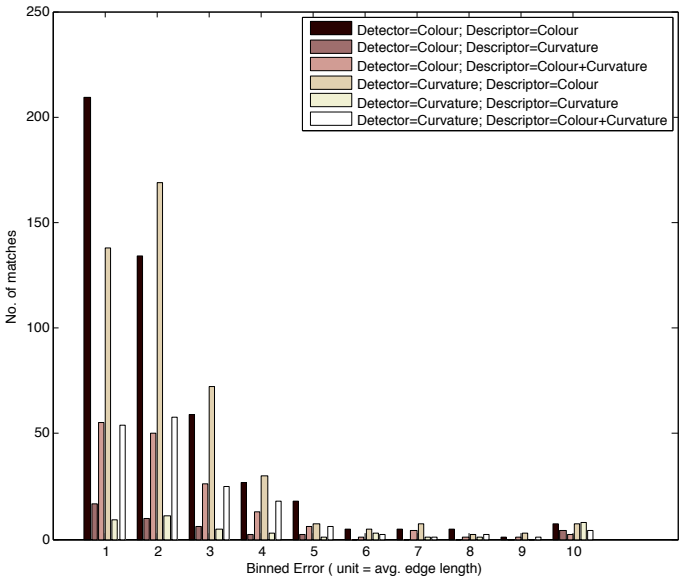
(a) Dino - Correspondances



(b) Dino - Erreurs



(c) Temple - Correspondances



(d) Temple - Erreurs

Figure 3.22: a) Résultats de mise-en-correspondance pour les cas ou la couleur a été utilisé pour la détection de points d'intérêt et pour le descripteur ; b) La distribution d'erreur quand différentes mesures de qualité ont été utilisées pour la détection de points d'intérêt et pour le descripteur.

(a) Mise-en-correspondance épars (b) Mise-en-correspondance dense

Figure 3.23: Résultats sur la séquence INRIA Dance-1: Trames 515 - 530

Chapter 4

Robust Probabilistic Factorization

In this chapter we address the problem of building a class of robust factorization algorithms that solve for the shape and motion parameters with both affine (weak perspective) and perspective camera models. We introduce a Gaussian/uniform mixture model and its associated EM algorithm. This allows us to address parameter estimation within a data clustering approach. We propose a robust technique that works with any affine factorization method and makes it resilient to outliers. In addition, we show how such a framework can be further embedded into an iterative perspective factorization scheme. We carry out a large number of experiments to validate our algorithms and to compare them with existing ones. We also compare our approach with factorization methods that use M-estimators.

4.1 Introduction

The problem of factorization can be formulated as the one of minimizing the following Frobenius norm:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\mathbf{S} - \hat{\mathbf{S}}(\boldsymbol{\theta})\|_F^2 \tag{4.1}$$

where matrix $\mathbf{S} = [\boldsymbol{s}_{ij}]$ denotes the measurement matrix containing matched 2-D image observations, $\hat{\mathbf{S}}(\boldsymbol{\theta}) = \mathbf{MP}$ denotes the prediction matrix that can be factorized into the *affine* motion matrix $\mathbf{M}$ and the *affine* shape matrix $\mathbf{P}$. Hence, we denote by $\boldsymbol{\theta}$ the affine motion **and** shape parameters collectively. In the error-free case, direct factorization of the observation matrix using SVD provides an optimal solution. More recently the problem of *robust* affine factorization has received a lot of attention and powerful algorithms that can deal with *noisy*, *missing*, and/or *erroneous* data were suggested.

Anandan & Irani [13] extended the classical SVD approach to deal with the case of directional uncertainty. They used the Mahalanobis norm instead of the Frobenius norm and they reformulated the factorization problem such that the Mahalanobis norm can be transformed into a Frobenius norm. This algorithm handles image observations with covariance up to a few pixels but it cannot cope with missing data, mismatched points, and/or outliers. More generally, a central idea is to introduce a weight matrix $\mathbf{W}$ of the same size as the measurement matrix $\mathbf{S}$. The minimization criterion then becomes:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\mathbf{W} \otimes (\mathbf{S} - \hat{\mathbf{S}}(\boldsymbol{\theta}))\|_F^2 \tag{4.2}$$

where $\otimes$ denotes the Hadamard product ($A = B \otimes C \iff a_{ij} = b_{ij}c_{ij}$) and $\mathbf{W} = [w_{ij}]$ is a matrix whose entries are weights that reflect the confidence associated with each image observation. The most common way of minimizing eq. (4.2) is to use alternation methods: these methods are based on the fact that, if either one of the matrices $\mathbf{M}$ or $\mathbf{P}$ is known, then there is a closed-form solution for the other matrix that minimizes eq. (4.2). Morris & Kanade [117] were the first to propose such an alternation method. The PowerFactorization method introduced by Hartley & Schaffalitzky [68], as well as other methods by Vidal & Hartley [162], and Brant [23] fall into this category. PowerFactorization is based on the PowerMethod for sparse matrix decomposition [61]. Notice that these techniques are very similar in spirit with PCA methods with missing data, Wiberg [165], Ikeuchi, Shum, & Reddy [145], Roweis [137], and Bishop [17]. Another way to alternate between the estimation of motion and shape is to use factor analysis, Gruber and Weiss [62], [63].

Alternatively, robustness may be achieved through *adaptive weighting*, i.e., by iteratively updating the weight matrix $\mathbf{W}$ which amounts to modifying the data $\mathbf{S}$, such as is done by Aanaes et al. [1]. Their method uses eq. (4.1) in conjunction with a robust loss function (see Stewart [150] and Meer [111] for details) to iteratively approximate eq. (4.2), getting a temporary optimum. The approximation is performed by modifying the original data $\mathbf{S}$ such that the solution to eq. (4.1) with *modified data* $\tilde{\mathbf{S}}$ is the same as the solution to eq. (4.2) with the original data:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\mathbf{W} \otimes (\mathbf{S} - \hat{\mathbf{S}}(\boldsymbol{\theta}))\|_F^2 = \arg\min_{\boldsymbol{\theta}} \|\tilde{\mathbf{S}} - \hat{\mathbf{S}}(\boldsymbol{\theta}))\|_F^2 \tag{4.3}$$

In [1] the weights are updated via IRLS [150]. This may well be viewed as both an iterative and an alternation method because the motion matrix $\mathbf{M}$ is estimated using SVD, then the shape matrix $\mathbf{P}$ is estimated knowing $\mathbf{M}$, while the image residuals are calculated and the data (the weights) are modified, etc. A similar algorithm that performs outlier correction was proposed by Huynh, Hartley, & Heyden [76]. Indeed, if the observations are noisy, the influence of outliers can be

decreased by iteratively replacing bad observations with "pseudo" observations. The convergence of such methods, as [1] or [76] is not proved but is tested through experiments with both simulated and real data.

An alternative to M-estimators are random sampling techniques developed independently in computer vision [46] and statistics [135] (see Meer [111] for a recent overview of these methods). For example, Huynh & Heyden [77] and Tardif et al. [155] use RANSAC, Trajkovic and Hedley use LMedS [157], and Hajder and Chetverikov [67] use LTS (Least Trimmed Squares) [136]. The major drawback of these methods is that they must consider a large number of subsets sampled from the observation matrix $\mathbf{S}$.

Generally speaking, robust regression techniques, such as the ones that we briefly discussed, work well in conjunction with affine factorization algorithms. Factorization was initially designed as a "closed-form solution" to multiple-view reconstruction, but robust affine factorization methods are iterative in nature, as explained above. This has several implications and some drawbacks. In the presence of a large number of outliers, proper initialization is required. The use of an influence function (such as the truncated quadratic) that tends to zero too quickly cause outliers to be ignored and hence, this raises the question of a proper choice of an influence function. The objective function is non-convex implying that IRLS will be trapped in local minima. The generalization of affine factorization to deal with perspective implies the estimation of depth values associated with each reconstructed point. This is generally performed iteratively [152], [32], [100], [101], [116], [122]. It is not yet clear at all how to combine *iterative robust methods* with *iterative projective/perspective factorization methods*.

In this chapter we cast the problem of robust factorization into the framework of data clustering [48]. Namely, we consider the problem of classifying the observed 2-D matched points into two categories: inliers and outliers. For that purpose we model the likelihood of the observations with a Gaussian/uniform mixture model. This leads to a maximum likelihood formulation with missing variables that can be solved with the EM algorithm [37], [110], [48]. Notice that this approach is different than the method proposed by Miller & Browning [115] requiring both labeled and ulabeled data sets.

We devise an EM algorithm within the framework of 3-D reconstruction and within the specific mixture model just outlined. Each EM step guarantees that the likelihood is increased. Hence EM may indeed lead to a local maximum of the likelihood. We show that in this particular case (normally distributed inliers and uniformly distributed outliers) the posterior probabilities have a very simple interpretation in terms of robust regression. We describe an affine factorization algorithm that uses EM; This algorithm is robust and it shares the convergence

properties just outlined. We also describe an extension of this algorithm to deal with the perspective camera model.

We performed several experiments in two different scenarios: multiple-camera calibration and 3-D reconstruction using turn-table data. Our method was compared to other methods on an equal footing: it performs as well as bundle adjustment to estimate external camera parameters. It performs better than IRLS (used in conjunction with the truncated quadratic) to eliminate outliers in some difficult cases.

The remainder of this chapter is organized as follows. Section 4.2 describes the probabilistic modelling of inliers and outliers using a mixture between a Gaussian and an uniform distribution. Section 4.3 explains how this probabilistic model can be used to derive an affine factorization algorithm and section 4.4 extends this algorithm to iterative perspective factorization. Sections 4.5 and 4.6 describe experiments performed with multiple-camera calibration and with multi-view reconstruction data sets. Section 4.8 compares our approach to M-estimators and section 4.9 draws some conclusions and gives some directions for future work.

4.2 Probabilistic modelling of inlier/outlier detection

The 2-D image points $\boldsymbol{s}_{ij}$ ($1 \leq i \leq k$, $1 \leq j \leq n$) are the observed values of an equal number of random variables s_{ij}. We introduce another set of random variables, z_{ij} which assign a category to each observation. Namely there are two possible categories, an *inlier* category and an *outlier* category. More specifically $z_{ij} =$ inlier means that the observation $\boldsymbol{s}_{ij}$ is an inlier while $z_{ij} =$ outlier means that the observation $\boldsymbol{s}_{ij}$ is an outlier.

We define the prior probabilities as follows. Let A_i be the area associated with image i and we assume that all the images have the same area, $A_i = A$. If an observation is an inlier, then it is expected to lie within a small circular image patch a of radius σ_0, $a = \pi\sigma_0^2$. The prior probability of an inlier is the proportion of the image restricted to such a small circular patch:

$$P(z_{ij} = \text{inlier}) = \frac{a}{A} \tag{4.4}$$

Similarly, if the observation is an outlier, its probability should describe the fact that it lies outside this small patch:

$$P(z_{ij} = \text{outlier}) = \frac{A - a}{A} \tag{4.5}$$

Moreover, an observation $\boldsymbol{s}_{ij}$, given that it is an inlier, should lie in the neighborhood of an estimation $\hat{\boldsymbol{s}}_{ij}$. Therefore, we will model the probability of an observation $\boldsymbol{s}_{ij}$ *given that it is assigned to the inlier category* with a Gaussian distribution centered on $\hat{\boldsymbol{s}}_{ij}$ and with a 2×2 covariance matrix $\mathbf{C}$. We obtain:

$$P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij}=\text{inlier}) = \frac{1}{2\pi(\det\mathbf{C})^{1/2}}\exp\left(-\frac{1}{2}d^2_{\mathbf{C}}(\boldsymbol{s}_{ij},\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\right) \tag{4.6}$$

where we denote by $d_{\mathbf{C}}$ the Mahalanobis distance:

$$d^2_{\mathbf{C}}(\boldsymbol{s}_{ij},\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) = (\boldsymbol{s}_{ij}-\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top\mathbf{C}^{-1}(\boldsymbol{s}_{ij}-\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{4.7}$$

Whenever the observation is an outlier, it may lie anywhere in the image. Therefore, we will model the probability of an observation $\boldsymbol{s}_{ij}$ *given that it is assigned to the outlier category* with a uniform distribution over the image area:

$$P(\boldsymbol{s}_{ij}|z_{ij}=\text{outlier}) = \frac{1}{A} \tag{4.8}$$

Since each variable z_{ij} can take only two values, marginalization is straightforward and we obtain:

$$\begin{aligned} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}) &= P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij}=\text{inlier})P(z_{ij}=\text{inlier}) + P(\boldsymbol{s}_{ij}|z_{ij}=\text{outlier})P(z_{ij}=\text{outlier}) \\ &= \frac{a}{2\pi(\det\mathbf{C})^{1/2}A}\exp\left(-\frac{1}{2}d^2_{\mathbf{C}}(\boldsymbol{s}_{ij},\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\right) + \frac{A-a}{A^2} \end{aligned} \tag{4.9}$$

We already defined the small area a as a disk of radius σ_0, $a=\pi\sigma_0^2$ and we assume that $a \ll A$. Using Bayes' formula[1], we obtain the posterior conditional probability of an observation to be an inlier. We denote this posterior probability by α^{in}_{ij}:

$$\alpha^{in}_{ij} = P_{\boldsymbol{\theta}}(z_{ij}=\text{inlier}|\boldsymbol{s}_{ij}) = \frac{1}{1+\frac{2}{\sigma_0^2}(\det\mathbf{C})^{1/2}\exp\left(\frac{1}{2}d^2_{\mathbf{C}}(\boldsymbol{s}_{ij},\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\right)} \tag{4.10}$$

The covariance matrix can be written as $\mathbf{C}=U\Lambda U^\top$ where U is a rotation and Λ is a diagonal form with entries λ_1 and λ_2. Hence $\det(\mathbf{C})=\lambda_1\lambda_2$. In order to plot and illustrate the shape of α^{in}_{ij} as a function of $\mathbf{C}$ we consider the case of an isotropic covariance, i.e., $\lambda_1=\lambda_2=\sigma^2$ and one may notice that the rotation becomes irrelevant in this case. We have: $\mathbf{C}=\sigma^2\mathbf{I}_2$. Eq. (4.10) writes in this case:

$$\alpha^{in}_{ij} = P_{\boldsymbol{\theta}}(z_{ij}=\text{inlier}|\boldsymbol{s}_{ij}) = \frac{1}{1+\frac{2\sigma^2}{\sigma_0^2}\exp\left(\frac{\|\boldsymbol{s}_{ij}-\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})\|^2}{2\sigma^2}\right)} \tag{4.11}$$

[1] $P(z_{ij}=\text{inlier}|\boldsymbol{s}_{ij})P(\boldsymbol{s}_{ij}) = P(\boldsymbol{s}_{ij}|z_{ij}=\text{inlier})P(z_{ij}=\text{inlier})$

This posterior probability is shown on Figure 4.1, i.e., the function $f_\sigma(x) = 1/(1+\sigma^2 \exp(x^2/2\sigma^2))$. Here σ takes discrete values in the interval $[0.05, 5]$ and $\sigma_0^2 = 2$, i.e., inliers lie within a circle of radius 2 pixels centered on a prediction. It is worthwhile to notice that, at the limit $\sigma \to 0$, we obtain a Dirac function:

$$f_0(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \tag{4.12}$$

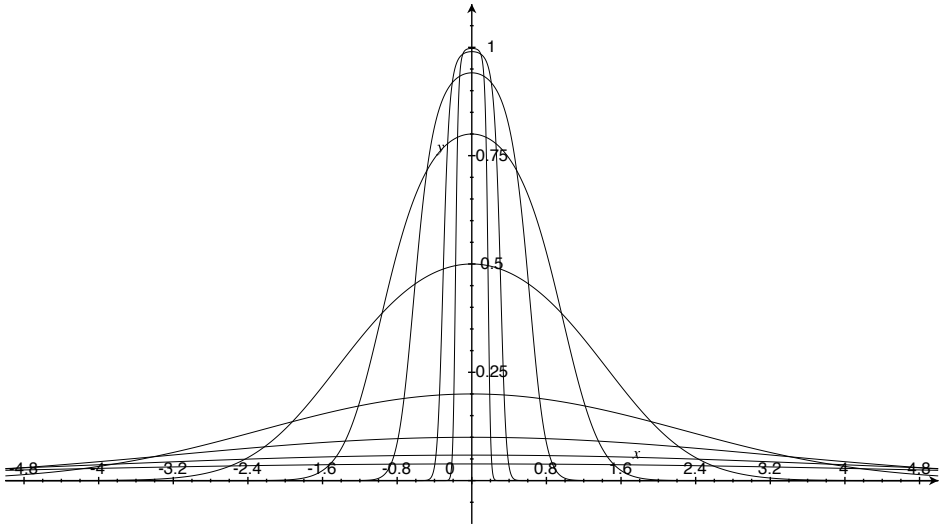


Figure 4.1: Various plots of the conditional posterior probability of an observation to be an inlier, i.e., $f_\sigma(x) = 1/(1+\sigma^2 \exp(x^2/2\sigma^2))$. This function corresponds to eq. (4.11) with $\sigma_0^2 = 2$. As the variance decreases, i.e., $\sigma = 5, 4, 3, 2, 1, 0.5, 0.25, 0.1, 0.05$, the function becomes more and more discriminant. It is worthwhile to notice that $\lim_{\sigma \to 0} f_\sigma(x)$ is a Dirac.

The posterior conditional probability of an observation to be an outlier is given by:

$$\alpha_{ij}^{out} = P_{\boldsymbol{\theta}}(z_{ij} = \text{outlier} | \boldsymbol{s}_{ij}) = 1 - \alpha_{ij}^{in} \tag{4.13}$$

4.2.1 Maximum likelihood with inliers

The maximum likelihood estimator (ML) maximizes the log-likelihood of the joint probability of the set of measurements, $P_{\boldsymbol{\theta}}(\mathbf{S})$. Assuming that the observations are independent and identically distributed, we have:

$$P_{\boldsymbol{\theta}}(\mathbf{S}) = \prod_{i,j} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}) \tag{4.14}$$

Since we assume that all the observations are inliers, eq. (4.9) reduces to:

$$P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}) = P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier}) \tag{4.15}$$

The log-likelihood of the joint probability becomes:

$$\log P_{\boldsymbol{\theta}}(\mathbf{S}) = -\frac{1}{2}\sum_{i,j}\left(d^2_{\mathbf{C}}(\boldsymbol{s}_{ij}, \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) + \log(\det \mathbf{C})\right) + \text{const} \tag{4.16}$$

which can be written as the following criterion:

$$Q_{ML} = \frac{1}{2}\sum_{i,j}\left((\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) + \log(\det \mathbf{C})\right) \tag{4.17}$$

The shape and motion parameters can be estimated by minimizing the above criterion with respect to $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{4.18}$$

Once an optimal solution is found, i.e., $\boldsymbol{\theta}^*$, it is possible to minimize eq. (4.17) with respect to the covariance matrix which yields (see appendix A.1):

$$\mathbf{C}^* = \frac{1}{m}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))^\top \tag{4.19}$$

where $m = kn$ is the total number of observations for k images and n 3-D points.

Alternatively, if one uses an isotropic covariance, i.e., $\mathbf{C} = \sigma^2\mathbf{I}$, By minimization of Q_{ML} with respect to $\boldsymbol{\theta}$ we obtain:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i,j}\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\|^2 \tag{4.20}$$

The optimal variance is given by (see appendix A.2):

$$\sigma^{2^*} = \frac{1}{2m}\sum_{i,j}\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))\|^2 \tag{4.21}$$

4.2.2 Maximum likelihood with inliers and outliers

In the presence of outliers, the previous method cannot be applied. Instead, one has to use the *joint probability* of the observations and of their assignments. Again, by assuming that the observations are independent, we have:

$$\begin{aligned} P_{\boldsymbol{\theta}}(\mathbf{S}, \mathbf{Z}) &= \prod_{i,j} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}, z_{ij}) \\ &= \prod_{i,j} P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij}) P(z_{ij}) \\ &= \prod_{i,j} \big(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier}) P(z_{ij} = \text{inlier})\big)^{\delta_{in}(z_{ij})} \\ &\quad \big(P(\boldsymbol{s}_{ij}|z_{ij} = \text{outlier}) P(z_{ij} = \text{outlier})\big)^{\delta_{out}(z_{ij})} \end{aligned} \tag{4.22}$$

The random variables $\delta_{in}(z_{ij})$ and $\delta_{out}(z_{ij})$ are defined by:

$$\delta_{in}(z_{ij}) = \begin{cases} 1 & \text{if} \quad z_{ij} = \text{inlier} \\ 0 & \quad\ \text{otherwise} \end{cases} \qquad \delta_{out}(z_{ij}) = \begin{cases} 1 & \text{if} \quad z_{ij} = \text{outlier} \\ 0 & \quad\ \text{otherwise} \end{cases}$$

By taking the logarithm of the above expression and grouping constant terms, we obtain:

$$\begin{aligned} \log P_{\boldsymbol{\theta}}(\mathbf{S}, \mathbf{Z}) &= \sum_{i,j} \big(\delta_{in}(z_{ij}) \log(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier})) \\ &+ \delta_{out}(z_{ij}) \log(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{outlier})) + \text{const}\big) \end{aligned} \tag{4.23}$$

This cannot be solved as previously because of the presence of the missing assignment variables z_{ij}. Therefore, they will be treated within an expectation-maximization framework. For this purpose we evaluate the *conditional expectation* of the log-likelihood over the random variables z_{ij}, given the observations $\mathbf{S}$:

$$\begin{aligned} E_Z\left[\log(P_{\boldsymbol{\theta}}(\mathbf{S}, \mathbf{Z}))|\mathbf{S}\right] &= \sum_{i,j} \big(\log(P_{\boldsymbol{\theta}}(\boldsymbol{s}_{ij}|z_{ij} = \text{inlier})) E_Z\left[\delta_{in}(z_{ij})|\mathbf{S}\right] \\ &+ \log(P(\boldsymbol{s}_{ij}|z_{ij} = \text{outlier})) E_Z\left[\delta_{out}(z_{ij})|\mathbf{S}\right]\big) \end{aligned} \tag{4.24}$$

In this formula we omitted the constant terms, i.e., the terms that do not depend on the parameters $\boldsymbol{\theta}$ and $\mathbf{C}$. The subscript Z indicates that the expectation is taken over the random variable z. From the definition of $\delta_{in}(z_{ij})$ we have:

$$\begin{aligned} E[\delta_{in}(z_{ij})] &= \delta_{in}(z_{ij} = \text{inlier})P(z_{ij} = \text{inlier}) + \delta_{in}(z_{ij} = \text{outlier})P(z_{ij} = \text{outlier}) \\ &= P(z_{ij} = \text{inlier}) \end{aligned}$$

Hence:

$$E_Z\left[\delta_{in}(z_{ij})|\mathbf{S}\right] = P(z_{ij} = \text{inlier}|\mathbf{S}) = P(z_{ij} = \text{inlier}|\boldsymbol{s}_{ij}) = \alpha_{ij}^{in}$$

and:

$$E_Z\left[\delta_{out}(z_{ij})|\mathbf{S}\right] = 1 - \alpha_{ij}^{in}$$

Therefore, after removing constant terms, the conditional expectation becomes:

$$E_Z\left[\log(P_{\boldsymbol{\theta}}(\mathbf{S},\mathbf{Z}))|\mathbf{S}\right] = -\frac{1}{2}\sum_{i,j}\alpha_{ij}^{in}\left(d^2_{\mathbf{C}}(\boldsymbol{s}_{ij},\hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) + \log(\det\mathbf{C})\right) \quad (4.25)$$

This leads to the following criterion:

$$Q_{EM} = \frac{1}{2}\sum_{i,j}\alpha_{ij}^{in}\left((\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top\mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}) + \log(\det\mathbf{C})\right) \quad (4.26)$$

4.3 Robust affine factorization with the EM algorithm

In this section we provide the details of how the robust affine factorization problem can be solved iteratively by maximum likelihood via the expectation-maximization (EM) algorithm [37], and how the observations can be classified into either inliers or outliers by maximum a-posteriori (MAP).

By inspection of equations (4.17) and (4.26) one may observe that the latter is a weighted version of the former and hence our formulation has strong simililarities with M-estimators and their practical solution, namely iteratively reweighted least-squres (IRLS) [150]. Nevertheless, the *weights* $\omega_{ij} = \alpha_{ij}^{in}$ were obtained using a Bayesian approach: they correspond to the posterior conditional probabilities of the observations (i.e., given that they are inliers), and such that the equality $\alpha_{ij}^{in} + \alpha_{ij}^{out} = 1$ holds for each observation. The structure and the shape of these posteriors are depicted by equations (4.10) and (4.11) and shown on Figure 4.1.

These probabilities are functions of the residual but they are parameterized as well by the 2×2 covariance matrix $\mathbf{C}$ associated with the normal probability distribution of the observations: One advantage of our formulation over IRLS is that this covariance is explicitly taken into consideration and estimated within the EM algorithm.

It is worthwhile to remark that the minimization of eq. (4.18) over the affine shape and motion parameters, i.e., $\boldsymbol{\theta}$, can be solved using an affine camera model and a factorization method such that the ones proposed in the literature [1, 68]. In practice we use the PowerFactorization method proposed in [68]. The minimization of eq. (4.26) can be solved in the same way, provided that estimates for the posterior probabilities α_{ij}^{in} are available. This can be done by iterations of the EM algorithm:

- The **E-step** computes the conditional expectation over the assignment variables associated with each observation, i.e., eq. (4.25). This requires a current estimate of both $\boldsymbol{\theta}$ and $\mathbf{C}$ from which the α_{ij}^{in}s are updated.
- The **M-step** maximizes the conditional expectation or, equivalently, minimizes eq. (4.26) with fixed posterior probabilities. This is analogous, but not identical, with finding the means $\boldsymbol{\mu}_{ij}$ and a common covariance $\mathbf{C}$ of $m = kn$ Gaussian distributions, with $\boldsymbol{\mu}_{ij} = \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})$. Nevertheless, the means $\boldsymbol{\mu} = \{\boldsymbol{\mu}_{11}, \ldots, \boldsymbol{\mu}_{kn}\}$ are parameterized by the global variables $\boldsymbol{\theta}$. For this reason, the minimization problem needs a specific treatment (unlike the classical mixture of Gaussians approach where the means are independent).

Therefore $\min_{\boldsymbol{\mu}} Q_{EM}$ in the standard EM method must be replaced by $\min_{\boldsymbol{\theta}} Q_{EM}$ and it does depend on $\mathbf{C}$ in this case:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i,j} \alpha_{ij}^{in} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) \tag{4.27}$$

Moreover, the covariance that minimizes eq. (4.26) can be easily derived from [17] and Appendix A.1:

$$\mathbf{C}^* = \frac{1}{\sum_{i,j} \alpha_{ij}^{in}} \sum_{i,j} \alpha_{ij}^{in} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))^\top \tag{4.28}$$

In many practical situations it is worthwhile to consider the case of an *isotropic covariance*, in which case the equations above reduce to:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i,j} \alpha_{ij}^{in} \|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))\|^2 \tag{4.29}$$

and

$$\sigma^{2^*} = \frac{1}{2\sum_{i,j}\alpha_{ij}^{in}} \sum_{i,j} \alpha_{ij}^{in} \|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))\|^2 \tag{4.30}$$

This may well be viewed as a special case of model-based clustering [48]. It was proved [110] that EM guarantees convergence, i.e., that $Q_{EM}^{q+1} < Q_{EM}^{q}$, where the overscript q denotes the q^{th} iteration, and that this implies the maximization of the joint probability of the observations: $P_{\boldsymbol{\theta}}(\mathbf{S})^{q+1} > P_{\boldsymbol{\theta}}(\mathbf{S})^{q}$. To conclude, the algorithm can be paraphrased as follows:

Affine factorization with EM:

Initialization: Use the PowerFactorization method to minimize eq. (4.18). This provides initial estimates for $\boldsymbol{\theta}$ (the affine shape and motion parameters). Estimate $\mathbf{C}$ (the covariance matrix) using eq. (4.19).

Iterate until convergence

Expectation: Update the values of α_{ij}^{in} according to eq. (4.10) or eq. (4.11).

Maximization: Minimize Q_{EM} over $\boldsymbol{\theta}$ (affine factorization) using either eq. (4.27) or eq. (4.29). Compute the covariance $\mathbf{C}$ with eq. (4.28) or the variance σ^2 with eq. (4.30).

Maximum a posteriori: Once the EM iterations terminate, choose in between inlier and outlier for each observation, i.e., $\max\{\alpha_{ij}^{in}; \alpha_{ij}^{out}\}$.

The algorithm needs initial estimates for the shape and motion parameters from which an initial covariance matrix can be estimated. This guarantees that, at the start of EM, all the residuals have equal importance. Nevertheless, "bad" observations will have a large associated residual and, consequently, the covariance is proportionally large. As the algorithm proceeds, the covariance adjusts to the current solution while the posterior probabilities α_{ij}^{in} become more and more discriminant as depticted on Figure 4.1. Eventually, observations associated with small residuals will be classified as inliers, and observations with large residuals will be classified as outliers.

The overall goal of 3-D reconstruction consists of the estimation of the shape and motion parameters: As just explained, we embed affine reconstruction in the M-step. Therefore, with our algorithm, robustness stays *outside* the factorization method at hand – is it iterative or not – and hence one can plug into EM any factorization procedure.

4.4 Robust perspective factorization

In this section we address the problem of 3-D reconstruction using instrinsically calibrated cameras. Moreover, we consider both the weak-perspective and the perspective camera models, and we explain how the affine solution provided by factorization can be upgraded to Euclidean reconstruction. We describe an algorithm that combines the EM affine factorization algorithm described above with an iterative perspective factorization algorithm [32, 173]. This results in a robust method for solving the 3-D Euclidean reconstruction problem as well as the multiple-camera calibration problem.

An image point $\boldsymbol{s} = (x, y)$ is the projection of a 3-D point $\tilde{\boldsymbol{X}}$:

$$x_{ij} = \frac{\mathbf{r}_i^x \cdot \tilde{\mathbf{X}}_j + t_i^x}{\mathbf{r}_i^z \cdot \tilde{\mathbf{X}}_j + t_i^z} = \frac{\mathbf{a}_i^x \cdot \tilde{\mathbf{X}}_j + b_i^x}{\varepsilon_{ij} + 1} \tag{4.31}$$

$$y_{ij} = \frac{\mathbf{r}_i^y \cdot \tilde{\mathbf{X}}_j + t_i^y}{\mathbf{r}_i^z \cdot \tilde{\mathbf{X}}_j + t_i^z} = \frac{\mathbf{a}_i^y \cdot \tilde{\mathbf{X}}_j + b_i^y}{\varepsilon_{ij} + 1} \tag{4.32}$$

We introduced the following notations: The rotation matrix $\mathbf{R}_i^\top = [\mathbf{r}_i^x \ \mathbf{r}_i^y \ \mathbf{r}_i^z]$ and the translation vector $\mathbf{t}_i^\top = (t_i^x \ t_i^y \ t_i^z)$ correspond to the motion parameters and they are also denoted the external camera parameters. Dividing the above equations with the *depth* t_i^z we obtain a similar set of scaled equations. We have: $\mathbf{a}_i^x = \mathbf{r}_i^x / t_i^z$, $\mathbf{a}_i^y = \mathbf{r}_i^y / t_i^z$, $b_i^x = t_i^x / t_i^z$ and $b_i^y = t_i^y / t_i^z$.

We denote by ε_{ij} the *perspective distorsion* parameters, namely the following ratios:

$$\varepsilon_{ij} = \frac{\mathbf{r}_i^z \cdot \tilde{\mathbf{X}}_j}{t_i^z} \tag{4.33}$$

Finally, the perspective equations, i.e., eqs. (4.31) and (4.32) can be written as:

$$\boldsymbol{s}_{ij}(1 + \varepsilon_{ij}) = \mathbf{A}_i \boldsymbol{X}_j \tag{4.34}$$

where $\boldsymbol{X} = (\tilde{\boldsymbol{X}}, 1)$ and $\mathbf{A}_i$ denotes the following 2×4 matrix:

$$\mathbf{A}_i = \begin{bmatrix} \boldsymbol{a}_i^x & b_i^x \\ \boldsymbol{a}_i^y & b_i^y \end{bmatrix} \tag{4.35}$$

From now on we can replace the parameter vector $\boldsymbol{\theta}$ with the affine shape and motion parameters, namely the point set $\mathcal{X} = \{\boldsymbol{X}_1, \dots \boldsymbol{X}_j, \dots \boldsymbol{X}_k\}$ and the matrix set $\mathcal{A} = \{\mathbf{A}_1, \dots \mathbf{A}_j, \dots \mathbf{A}_n\}$. Using these notations, eq. (4.27) can now be written as:

$$\min_{\mathcal{A},\mathcal{X}} \frac{1}{2} \sum_{i,j} \alpha_{ij}^{in} (\boldsymbol{s}_{ij}(1+\varepsilon_{ij}) - \mathbf{A}_i \boldsymbol{X}_j)^\top \mathbf{C}^{-1} (\boldsymbol{s}_{ij}(1+\varepsilon_{ij}) - \mathbf{A}_i \boldsymbol{X}_j) \tag{4.36}$$

which can be solved via the EM affine factorization algorithm with $\varepsilon_{ij} = 0, \forall (i,j)$. A weak-perspective camera model can then be used for upgrading to Euclidean reconstruction.

The introduction of the perspective camera model adds non null perspective-distorsion parameters ε_{ij}, i.e., eq. (4.33). One fundamental observation is the following: if estimates for the parameters $\varepsilon_{ij}, \forall i \in [1 \dots k], \forall j \in [1 \dots n]$ are available, then this corresponds to a weak-perspective camera model that is closer to the true perspective model. If the true values of the perspective-distortion parameters are available, the corresponding weak-perspective model corresponds exactly to the perspective model. Hence, the problem reduces to affine factorization followed by Euclidean upgrade. Numerous iterative algorithms have been suggested in the literature for estimating the perspective-distortion parameters associated with each 2-D observation, both with uncalibrated and calibrated cameras [152], [32], [100], [101], [116], [122] to cite just a few. One possibility is to perform *weak-perspective iterations*. Namely, the algorithm starts with a *zero perspective distorsion* (or weak-perspective approximation) and then, at each iteration, it updates the perspective distorsions using eq. (4.33). To conclude, the robust perspective factorization algorithm can be summarized as follows:

Robust perspective factorization:

Initialization: Set $\varepsilon_{ij} = 0, \forall i \in [1 \dots k], \forall j \in [1 \dots n]$. Use the same initialization step as the **affine factorization with EM** algorithm.

Iterate until convergence:

Affine factorization with EM: Iterate until convergence the E- and M-steps of the algorithm described in the previous section.

Euclidean upgrade: Recover the rotations, translations, and 3-D Euclidean coordinates from the affine shape and affine motion parameters.

Perspective update: Estimate new values for the parameters $\varepsilon_{ij}, \forall i \in [1 \dots k], \forall j \in [1 \dots n]$. If the current depth values are identical with the previously estimated ones, then terminate, else iterate.

Maximum a posteriori: After convergence choose in between inlier and outlier for each observation, i.e., $\max\{\alpha_{ij}^{in}; \alpha_{ij}^{out}\}$.

4.5 Multiple-camera calibration

In this section we describe how the solution obtained in the previous section is used within the context of multiple-camera calibration. As already described above, we are interested in the estimation of the external camera parameters, i.e., the alignment between a global reference frame (or the calibration frame) and the reference frame associated with each one of the cameras. We assume that the internal camera parameters were accurately estimated using available software. There are many papers available that address the problem of internal camera calibration either from 3-D reference objects [45], 2-D planar objects [175], 1-D objects [176] or self-calibration, e.g., from point-correspondences [98], [69], [99].

Figure 4.2 shows a partial view of a multiple-camera setup as well as the one-dimensional object used for calibration. In practice we used three different camera configurations as depicted in Figure 4.4: two 30 camera configurations and one 10 camera configuration. These camera setups will be referred to as the *Corner Case*, the *Arc Case*, and the *Semi-Spherical Case*. Finding point correspondences accross the images provided by such a setup is an issue in its own right because one has to solve for a multiple wide-baseline point correspondence problem. We will briefly describe the practical solution that we retained and which maximizes the number of points that are matched over all the views. Nevertheless, in practice there are missing observations as well as badly detected image features, bad matches, etc. The problem of missing data has already been addressed. Here we concentrate on the detection and rejection of outliers.

We performed multiple camera calibration with two algorithms: The robust perspective factorization method previously described and bundle adjustment. We report a detailed comparison between these two methods. We further compare our robust method with a method based on M-estimators.

(a) (b)

Figure 4.2: (a): Partial view of a 30-camera setup. (b): The calibration data are gathered by moving a one-dimensional object in the common field of view of the cameras.

As already mentioned, we use a simple 1-D object composed of four identical markers with known 1-D coordinates. These coordinates form a projective-invariant signature (the cross-ratio) that is used to obtain 3-D to 2-D matches between the markers and their observed image locations. With finely syncronized cameras it is possible to gather images of the object while the latter is freely moved in order to cover the 3-D space that is commonly viewed by all cameras. In the three examples below we used 73, 58, and 32 frames, i.e., 292, 232, and 128 3-D points. The number of cameras in each setup is 30, 30 and 10, respectively. Therefore, in theory there should be 8760, 6960, and 1280 2-D observations.

Figure 4.3 depicts three possible image configurations: (a) four distinct connected components that correspond without ambiguity to the four markers, (b) a degenerate view of the markers, due to strong perspective distorsion, that results in a number of connected components that cannot be easily matched with the four markers, and (c) only two connected components are visible in which case one cannot establish a reliable match with the four markers. In practice we perform a connected-component analysis that finds the number of blobs in each image. Each such blob is characterized by its center and second order moments, i.e., Figure 4.3 (d), (e), and (f). These blobs are matched with the object markers. In most of the cases the match is unambiguous, but in some cases a blob may be matched with several markers.

Let as before, $\boldsymbol{s}_{ij}$ denote the center of a blob from image i that matches marker j. The second order moments of this blob can be used to compute an initial 2×2 covariance matrix $\mathbf{C}_{ij}^{0}$ for each such observation. Moreover, we introduce a binary variable, μ_{ij}, which is equal to 0 if the observation s_{ij} is missing and equal to 1

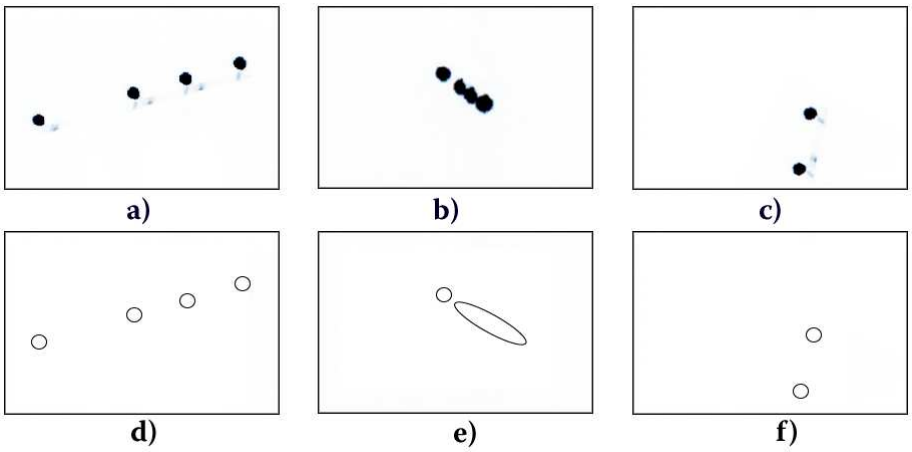


Figure 4.3: Top: These are typical images where the number of connected components depend on the position and orientation of the calibrating object with respect to the cameras. Bottom: Detected blobs with their centers and associated covariance, i.e., second-order moments.

otherwise. The multiple-camera calibration algorithm can now be paraphrased as follows:

Multiple camera calibration:

Initialization: Use eq. (4.18) to estimate the current affine shape and motion parameters in the presence of some missing data:

$$\boldsymbol{\theta}^q = \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i,j} \mu_{ij} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top (\mathbf{C}_{ij}^0)^{-1} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))$$

Estimate the initial covariance matrix using eq. (4.19):

$$\mathbf{C}^q = \frac{1}{m} \sum_{i,j} \mu_{ij} (\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^q))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^q))^\top$$

Set $\varepsilon_{ij} = 0, \forall i \in [1 \ldots k], \forall j \in [1 \ldots n]$

Iterate until convergence:

Affine factorization with EM: Iterate until convergence the E- and M-steps of the algorithm described in section 4.3.

Euclidean upgrade: Recover the rotations, translations, and 3-D Euclidean coordinates from the affine shape and affine motion parameters.

Perspective update: Estimate new values for the parameters $\varepsilon_{ij}, \forall i \in [1 \dots k], \forall j \in [1 \dots n]$. If the current depth values are identical with the previously estimated ones, terminate, else iterate.

Both the initialization and the M steps of the above algorithm perform affine factorization in the presence of uncertainty and missing data. In [173] we compared several such algorithms and we came to the conclusion that the PowerFactorization [68] algorithm outperforms the other tested algorithms. This is what we will be using in order to solved the affine factorization in the presence of uncertainty and missing data. For more details about the method, please consult Appendix A.3.

In order to assess quantitatively the performance of our algorithm, we compared it with an implementation of the bundle adjustment method along the lines described in [69]. This comparison requires the estimation of the rotations and translations allowing the alignment of the two reconstructed 3-D sets of points with the cameras. We estimate these rotations and translations using a set of control points. Indeed, both the robust perspective factorization and the bundle adjustment algorithms need a number of control points with known Euclidean 3-D coordinates. In practice, the calibration procedure provides such a set. This set of control points allows one to define a global reference frame. Let $\boldsymbol{P}_j^c$ denote the 3-D coordinates of the control points estimated with our algorithm, and let $\boldsymbol{Q}_j^c$ denote their 3-D coordinates provided in advance. Let λ, $\mathbf{R}$, and $\boldsymbol{t}$ be the scale, rotation and translation allowing the alignment of the two sets of control points. We have:

$$\min_{\lambda,\mathbf{R},\boldsymbol{t}} \sum_{j=1}^{8} \|\lambda \mathbf{R} \boldsymbol{Q}_j^c + \boldsymbol{t} - \boldsymbol{P}_j^c\|^2 \tag{4.37}$$

The minimizer of this error function can be found in closed form either with unit quaternions [74] to represent the rotation $\mathbf{R}$ or with dual-number quaternions [164]

to represent the rigid motion $\mathbf{R}, \boldsymbol{t}$. Similarly, one can use the same procedure to estimate the scale λ', rotation $\mathbf{R}'$, and translation $\boldsymbol{t}'$ associated with the 3-D reconstruction obtained by bundle adjustment.

Finally, in order to evaluate the quality of the results we estimated the following measurements:

The 2D error is measured in pixels and corresponds to the RMS error between the observations and the predictions weighted by their posterior probabilities:

$$\left(\frac{\sum_{i,j} \alpha_{ij}^{in} \| \boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*) \|^2}{\sum_{i,j} \alpha_{ij}^{in}} \right)^{1/2}$$

The 3D error is measured in milimeters and corresponds to the RMS error between the two sets of 3-D points obtained with our algorithm and with bundle adjustment:

$$\left(\frac{\sum_{j=1}^{n} \| \boldsymbol{P}_j - \boldsymbol{P}'_j \|^2}{n} \right)^{1/2}$$

The error in rotation is measured in degrees and depicts the average angular error of the rotation matrices over all the cameras. With the same notations as before, let $\mathbf{R}_i$ and $\mathbf{R}'_i$ be the rotations of camera i as obtained with our algorithm and with bundle adjustment. Let $\boldsymbol{v}$ be an arbitrary 3-D vector. The dot product $(\mathbf{R}_i \boldsymbol{v}) \cdot (\mathbf{R}'_i \boldsymbol{v})$ is a reliable measure of the cosine of the angular discrepancy between the two estimations. Therefore the RMS error in rotation can be measured by the angle:

$$\arccos \left(\frac{180}{\pi} \sqrt{\frac{\boldsymbol{v}^\top \left(\sum_{i=1}^{k} \mathbf{R}_i^\top \mathbf{R}'_i \right) \boldsymbol{v}}{k}} \right)$$

The error in translation is measured in milimeters with:

$$\left(\frac{\sum_{i=1}^{k} \| \boldsymbol{t}_i - \boldsymbol{t}'_i \|^2}{k} \right)^{1/2}$$

As already mentioned, we used three camera setups. All setups use identical 1024×768 Flea cameras from Point Grey Research Inc.[2] The intrinsic parameters were estimated in advance. Two of the setups use 30 cameras, whereas the third

[2] http://www.ptgrey.com/products/flea/

one uses 10 cameras. We denote these setups as *Corner Case*, *Arc Case* and *Semi-Spherical Case*, based on the camera layout.

The results are summarized on Figures 4.4 and 4.5 and on Tables 4.1 and 4.2. Let us analyze in more detail these results. In the *Corner Case* there are 30 cameras and 292 3-D points. Hence, there are 8760 possible predictions out of which only 5527 are actually observed, i.e., 36% predictions correspond to missing 2-D data. The algorithm detected 5202 2-D inliers. An inlier is an observation with a posterior probability greater than 0.4. Next, the outliers are marked as missing data. Eventually, in this example, 285 3-D points were reconstructed (out of a total of 292) and all the cameras were correctly calibrated. The number of iterations of the robust perspective factorization algorithm (refered to as affine iterations) is equal to 7. On an average, there were 2.4 iterations of the EM algorithm. The obtained reconstruction has a smaller 2-D reprojection error (0.30 pixels) than the one obtained by bundle adjustment (0.58 pixels).

In any of the 3 calibration scenarios, the proposed method outperforms bundle adjustment results, as it can be observed in Table 4.2. This is in part due to the fact that the bundle adjustment algorithm does not have a mechanism for outlier rejection.

Figure 4.5 shows the evolution of the algorithm as it iterates from a weak-perspective solution to the final full-perspective solution. At convergence, the solution found by our method (shown in blue or dark in the absence of colors) is practically identical to the solution found by bundle adjustment (which is shown in grey).

Multi-camera calibration		*Corner Case*	*Arc Case*	*Semi-Spherical Case*
Input	# Cameras	30	30	10
	# 3-D Points	292	232	128
	# 2-D Predictions	8760	6960	1280
	# Missing observations	36%	0%	33%
	# 2-D Observations	5527	6960	863
Results	# 2-D Inliers	5202	6790	784
	# 3-D Inliers	285	232	122
	2D error (pixels)	0.30	0.19	0.48
	3D error (mm)	6.91	2.65	4.57
	Rot. error (degrees)	0.13	0.18	0.27
	Tr. error (mm)	27.02	9.37	24.21
	# Aff. iter. (# EM iter.)	7 (2.4)	11 (2)	8 (3.2)

Table 4.1: Summary of the camera calibration results for the three setups.

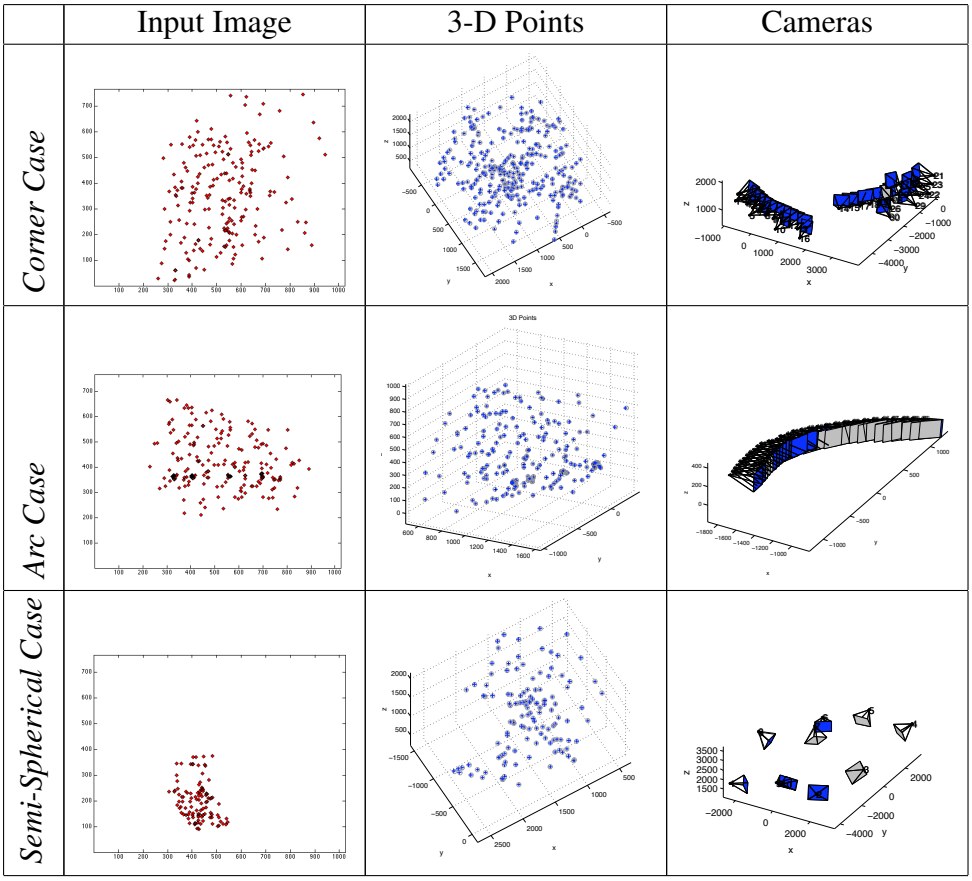


Figure 4.4: Multiple camera calibration results. Left: A typical set of 2-D observations associated with one camera. Middle: Reconstructed 3-D points with our method (blue) and with bundle adjustment (grey). Right: Camera calibration results obtained with our method (blue) and with bundle adjustment (grey) .

Multi-camera calibration	*Corner Case*	*Arc Case*	*Semi-Spherical Case*
Proposed Method	0.30	0.19	0.48
Bundle Adjustment	0.58	0.61	0.95

Table 4.2: Comparison between the proposed method and bundle adjustment for the three camera calibration setups. The error is measured in pixels and represents the 2D error.

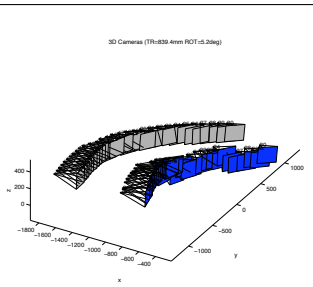
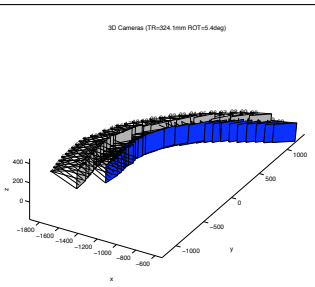
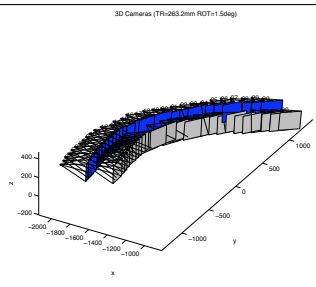
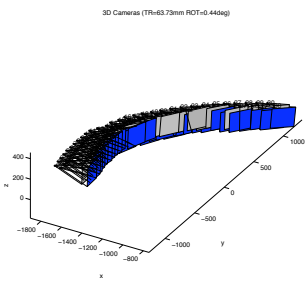
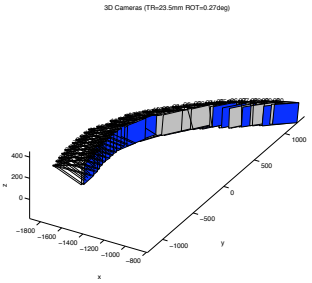
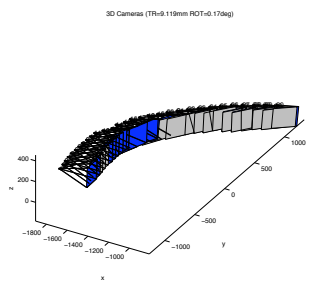
1^{st} iteration	2^{nd} iteration	3^{rd} iteration
		
2D Err. = 59.32	2D Err.= 18.43	2D Err. =15.92
4^{th} iteration	5^{th} iteration	8^{th} iteration
		
2D Err. = 4.53	2D Err. = 1.61	2D Err. = 0.24

Figure 4.5: Iterations of the robust perspective factorization algorithm in the *Arc Case* and comparison with bundle adjustment. The first iteration corresponds to weak-perspective factorization. The bundle adjustment solution is shown in grey. The bundle adjustment solution does not perform outlier treatment.

4.6 3-D Reconstruction

The robust perspective factorization algorithm was also applied to the problem of 3-D reconstruction from multiple views. For this purpose we used images of objects using a single camera and a turning table. More specifically, we used the following data sets:

- The "Dino" and "Temple" data sets from the Middlebury's evaluation of Multi-View Stereo reconstruction algorithms;[3]
- The "Oxford dinausor" data set,[4] and
- The "Square Box" data set.

We used the OpenCV[5] pyramidal implementation of the Lucas & Kanade interest point detector and tracker [21] to obtain the initial set of matched 2-D observations. This provides the $2k \times n$ measurement matrix $\mathbf{S}$ as well as the missing-data binary variables μ_{ij} associated with each observation. Figure 4.6 and Table 4.3 summarize the camera calibration and reconstruction results. For both the Middlebury data sets (Dino and Temple) and for the Oxford data set (Dinausor) we compared our camera calibration results with the calibration data provided with the data sets, i.e., we measured the error in rotation and the error in translation between our results and the data provided in advance.

3-D reconstruction		Dino	Temple	Box	Dinausor
Input	# Views	12	16	64	36
	Size of $\mathbf{S}$ matrix	24×480	32×758	128×560	72×1516
	# 2-D predictions	5760	12128	35840	54576
	% Missing observations	11%	21%	17%	85%
	# 2-D Observations	5140	9573	29733	8331
Results	# 2-D Inliers	3124	6811	25225	7645
	# 3-D Inliers	370	720	542	1437
	2D error (pixels)	0.82	0.93	0.69	0.33
	Rot. error (degrees)	1.49	2.32	–	0.00
	Trans. error (mm)	0.01	0.01	–	0.00
	# Aff. iter. (# EM iter.)	7 (4)	7 (3.14)	9 (3)	7 (3.29)

Table 4.3: Summary of the 3-D reconstruction results for the four data sets.

[3] http://vision.middlebury.edu/mview/data/
[4] http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
[5] http://www.intel.com/technology/computing/opencv/

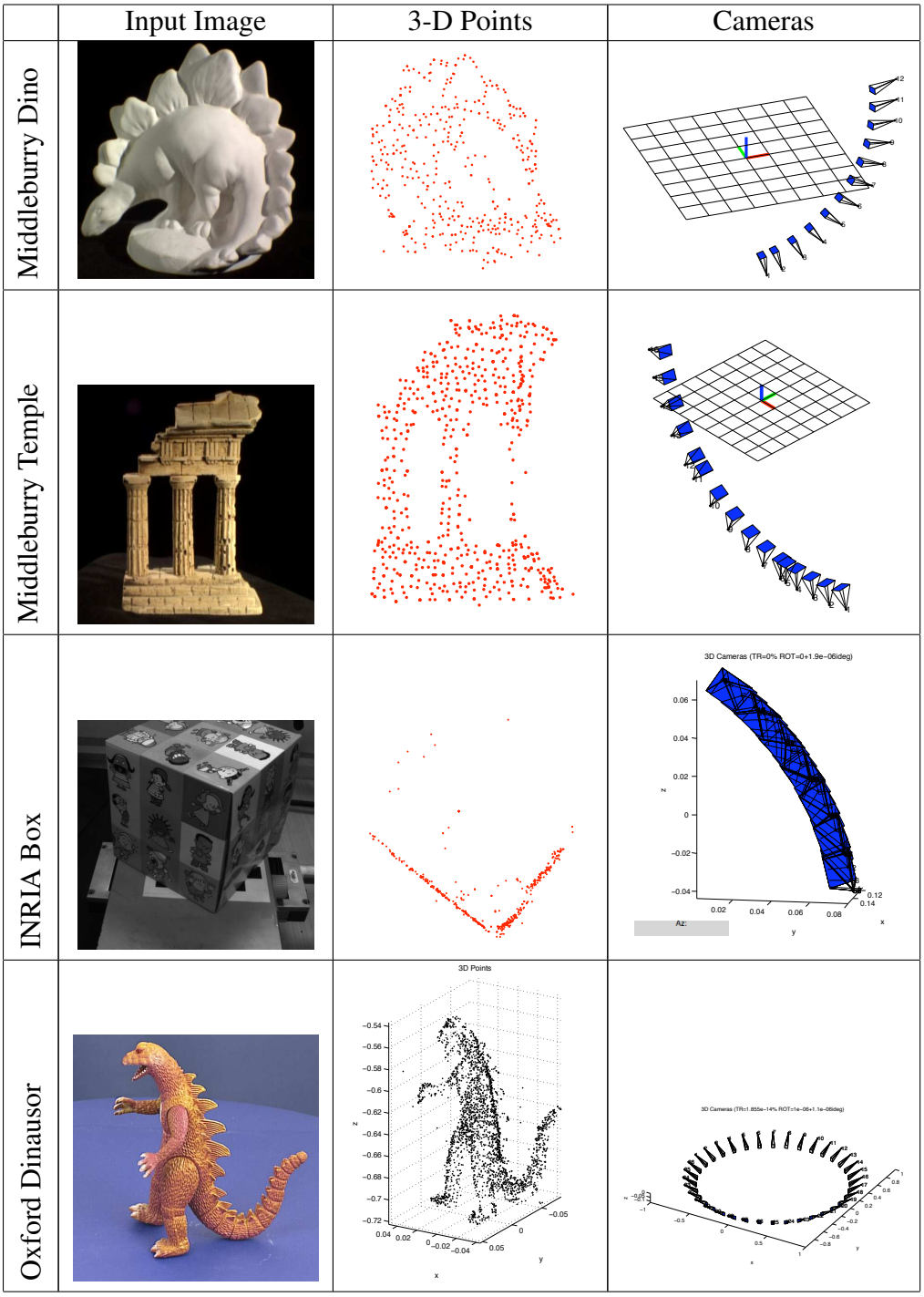


Figure 4.6: Groundtruth data is represented in gray (light) colour, whereas reconstruction results are represented in blue (dark) colour.

The first row in Table 4.3 introduces the test cases. The second row corresponds to the number of views. The third and forth rows provide the size of the measurement matrix based on the number of views and on the maximum number of observations over all the views. The sixth row provides the number of actually observed 2-D points while the seventh row provides the number of 2-D inliers (observations with a posterior probability greater than 0.4). The eighth row provides the number of actual 3-D reconstructed points. One may notice that, in spite of missing data and of the presence of outliers, the algorithm is able to reconstruct a large percentage of the observed points. In the "Dino" and "Temple" example we compared our camera calibration results with the groundtruth calibration parameters provided with the Middlebury multi-stereo dataset. Please note that these datasets are made available without the groundtruth 3-D data. They are typically used by the community to compare results for 3-D dense reconstructions. The rotation error stays within 3 degrees. The translation error is very small because, in this case we aligned the camera centers and not the 3-D coordinates of some reconstructed points.

The results obtained with the Oxford "Dinausor" need some special comments. Because of the very large percentage of missing data, we have been unable to initialize the solution with the PowerFactorization method. Therefore, we provided the camera calibration parameters for initialization. However, this kind of problem can be overcome by using an alternative affine factorization algorithm [155].

4.7 Comparison between different affine factorization algorithms

We have run several tests in order to compare the performances of different existing factorization algorithms. Before providing the list of algorithm, we would like to make a small parenthesis, explaining the differences between a rank-3 and a rank-4 formulation for the factorization problem.

Let us recall the equations (4.34) and (4.35) combined:

$$\boldsymbol{s}_{ij}(1+\varepsilon_{ij}) = \underbrace{\begin{bmatrix} \boldsymbol{a}_i^x & b_i^x \\ \boldsymbol{a}_i^y & b_i^y \end{bmatrix}}_{\mathbf{A}_{i(2\times 4)}} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{X}} \\ 1 \end{bmatrix}}_{\mathbf{X}_{(4\times 1)}}$$

We call this a a rank-4 factorization since, since the first matrix contains 4 columns. When stacking all the measurements across all cameras we obtain matrices $\mathcal{S} = \mathcal{A}\mathcal{X}$, where the last row of matrix $\mathcal{X}$ is all 1. This is a constraint that

has to be explicitly taken into account by the factorization scheme and thus, not all methods can provide it.

In order to obtain a rank-3 factorization, the image projections $\boldsymbol{b}_i = (b_i^x, b_i^y)$ of the world origin have to be subtracted from the point correspondences. This leads to:

$$\boldsymbol{s}_{ij}(1+\varepsilon_{ij}) - \boldsymbol{b}_i = \underbrace{\begin{bmatrix} \boldsymbol{a}_i^x \\ \boldsymbol{a}_i^y \end{bmatrix}}_{\mathbf{A}_{i(2\times 3)}} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{X}} \end{bmatrix}}_{\mathbf{X}_{(3\times 1)}}$$

In practice, it means that one needs to estimate the projections $\boldsymbol{b}_i = (b_i^x, b_i^y)$ of the world origin. That is typically not known in advance and it is done by taking the average of all the projected points in each image. However, if not all points are projecting in all the images, the estimation becomes less accurate. Therefore, a rank-4 factorization is desirable, if the factorization algorithm can account for the additional constraint.

We have tested a number of factorization algorithms within the proposed robust framework, both in the robust affine and the robust perspective factorization setting. The algorithms tested are the following:

- **SVD**: the original algorithm proposed by Tomasi et al. [156], based on singular value decomposition. Since the algorithm does not support missing data, the empty entries are replaced with the average of the existing ones. This is a rank-3 factorization algorithm.

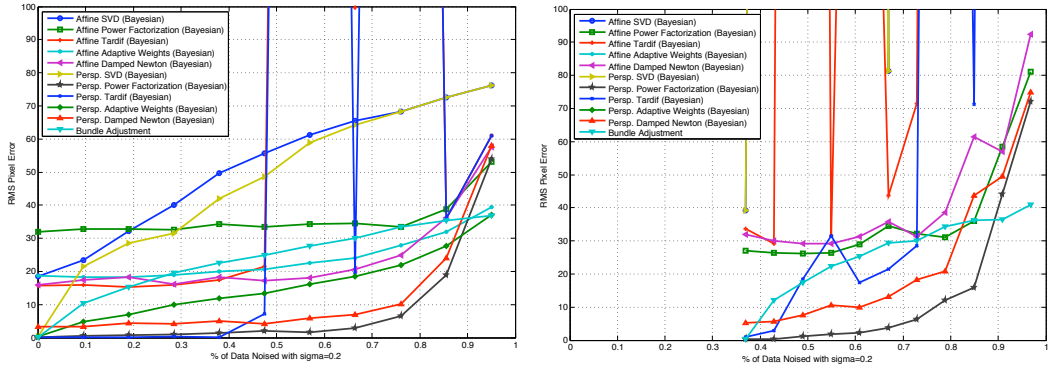
- **Power Factorization**: the Power Factorization with Uncertainty algorithm, proposed by Hartley [68], also detailed in the Appendix A.3. This is a rank-4 factorization algorithm.

- **Tardif**: the batch matrix factorization algorithm proposed by Tardif et al. [155]. This is a rank-4 factorization algorithm.

- **Adaptive Weights**: the robust factorization algorithm proposed by Aanæs et al. [1]. This is a rank-3 factorization algorithm.

- **Damped Newton**: the Damped Newton algorithm proposed by Buchanan et al. [29]. This is a rank-3 factorization algorithm. Even though it is considered a factorization algorithm, it is not a linear method. The minimization contains a non-linear step.

- **Bundle Adjustment**: a sparse bundle adjustment implementation, provided by Lourakis et at. [96]. This is not a factorization algorithm, but a non-linear minimization technique. Since the bundle adjustment needs an initial solution, it was always initialized using the affine PowerFactorization method.



(a) *Arc Case* (b) *Corner Case*

Figure 4.7: Behavior of various factorization algorithms when an increasing percentage of the input data are corrupted by high-amplitude noise, namely $\sigma = 0.2$ of the image size.

We present the results obtained in the *Arc Case* and *Corner Case* in Figure 4.7, where we have progressively noised the matches with Gaussian noise ($\sigma = 0.20$ of the image size). The PowerFactorization provides the best overall results among the chosen methods. It is our method of choice used when presenting any of the other results in this chapter.

4.8 Comparison with other methods

As already mentioned in section 4.3, our robust ML estimator has strong similarities with M-estimators and their practical implementation, i.e., IRLS [150]. Previous work on robust affine factorization has successfully used the following reweighting function ϕ that corresponds to the truncated quadratic:

$$\phi(x) = \begin{cases} 1 & \text{if } |x| < k \\ \sqrt{\frac{k^2}{x^2}} & \text{otherwise} \end{cases} \tag{4.38}$$

It is therefore tempting to replace the EM procedure of our algorithm with an IRLS procedure, which amounts to replace the posterior probabilities of inliers

α_{ij}^{in} given by eq. (4.10) with the weights ϕ_{ij} given by eq. (4.38). The latter tends to zero most quickly allowing aggressive rejection of outliers. One caveat is that the efficiency of IRLS depends on the tuning parameter k. Unfortunately, the latter cannot be estimated within the minimization process as is the case with the covariance matrix. However, we noted that the results that we obtained do not depend on the choice of k. In all the experiments reported below, we used $k = 1$, The plot of the truncated quadratic for different k values is plotted in Figure 4.8.

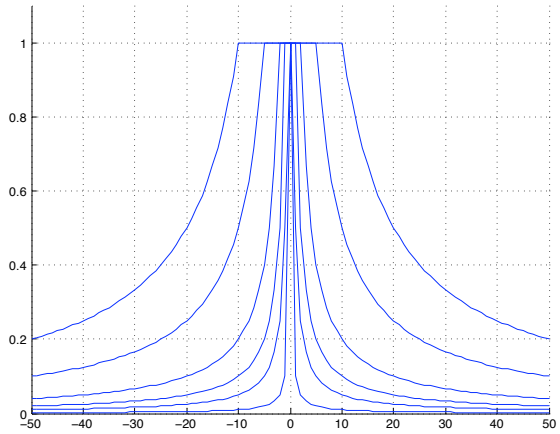


Figure 4.8: Reweighting function for $k = 0.1, 0.5, 1, 2, 5, 10$.

We compared the two robust methods (our EM-based robust perspective factorization algorithm and an equivalent IRLS-based algorithm) with five data sets for which we had the ground truth: Three multiple-camera calibrations data sets (the *Corner Case*, the *Arc Case* and the *Semi-Spherical Case*) and two multi-view reconstruction data sets (*Dino* and *Temple*). The results of this comparison are summarized in Table 4.4.

In the *Corner* case the quality of the results are very similar: our algorithm accepted 94% of the total number of observations as inliers and reconstructed 99.3% of the total number of 3-D points, while IRLS accepted all the observations as inliers and reconstructed all the 3-D points. Similar results are obtained in the *Arc* and *Semi-Spherical* cases, where the proposed method performs slightly better. Both algorithms were able to reconstruct the *Dino* and the *Temple*, but our algorithm yields more accurate results. Outlier detection is summarized in Table 4.5.

A more thorough comparison with robust as well as non robust 3-D reconstruction methods is provided in Figure 4.9. The proposed algorithm is denoted by "Persp. Power Factorization (Bayesian)", while the IRLS method is named "Persp. Power Factorization (IRLS - Truncated Quadratic)" and the non-robust

Dataset	Method	2-D Inliers	3-D Inliers	2-D err.	3-D err.	Rot. err.	Trans. err.
Corner	EM	5202 (5527)	285 (292)	**0.30**	6.91	0.13	27.02
	IRLS	5526 (5527)	288 (292)	0.40	6.91	0.14	26.61
Arc	EM	6790 (6960)	232 (232)	**0.19**	2.65	0.18	9.37
	IRLS	6960 (6960)	232 (232)	0.22	2.54	0.16	8.78
Semi-Spherical	EM	784 (863)	122 (128)	**0.48**	4.57	0.27	24.21
	IRLS	862 (863)	128 (128)	0.62	4.66	0.29	23.91
Dino	EM	3124 (5140)	370 (480)	**0.82**	–	1.49	0.01
	IRLS	3411 (5140)	390 (480)	2.57	–	2.13	0.01
Temple	EM	6811 (9573)	720 (758)	**0.93**	–	2.32	0.01
	IRLS	7795 (9573)	731 (758)	1.69	–	2.76	0.03

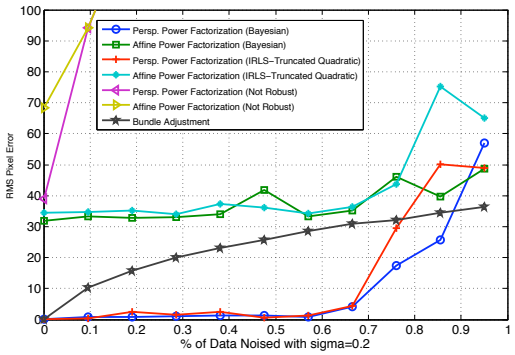
Table 4.4: Comparison between robust perspective factorization results using EM and IRLS. The figures in paranthesis correspond to the total number of observations (third column) and to the total number of expected 3-D points (fourth column).

	Corner	*Arc*	*Semi-Spherical*	*Dino*	*Temple*
EM	6%	2%	9%	39%	29%
IRLS	0%	0%	0%	34%	19%

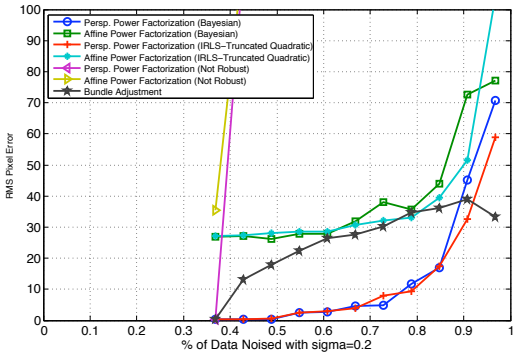
Table 4.5: Percentage of outliers detected by the two algorithms.

method is called "Persp. Power Factorization (Not Robust)". Affine factorization algorithms are also presented, together with the results of bundle adjustment. The bundle adjustment method was always initialized using the PowerFactorization method. The robust perspective factorization method proposed in this chapter is the most resilient to high-amplitude noise. It generally performs better than the IRLS method and provides a clear advantage against the non-robust methods, which *exit* the graphs as soon as the noise level increases. As it can be observed, in the *Semi-Spherical Case*, the solution deteriorates a lot faster in the presence of noise, due to the lack of the redundancy in the data (128 3-D points and 10 cameras, versus 292 points and 30 cameras in the *Corner Case* and 232 points and 30 cameras in the *Arc Case*).

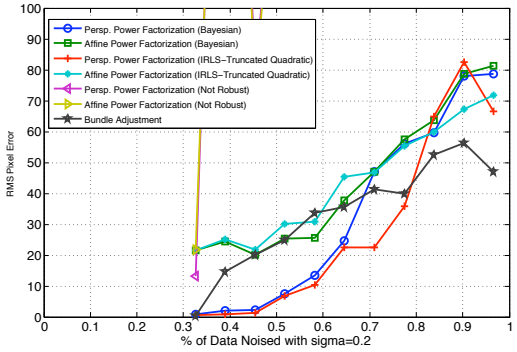
Figure 4.10 compares our method (a), with the bundle adjustment method (b), in the *Arc Case*and when 20% of the input data was corrupted by high-amplitude noise ($\sigma = 0.20$ of the image size). On both figures the ground truth is shown in grey and the result of the algorithm is shown in blue (or dark in the absence of color). Notice that with this level of data perturbation, bundle adjustment completely failed to find the correct solution.



(a) *Arc Case*

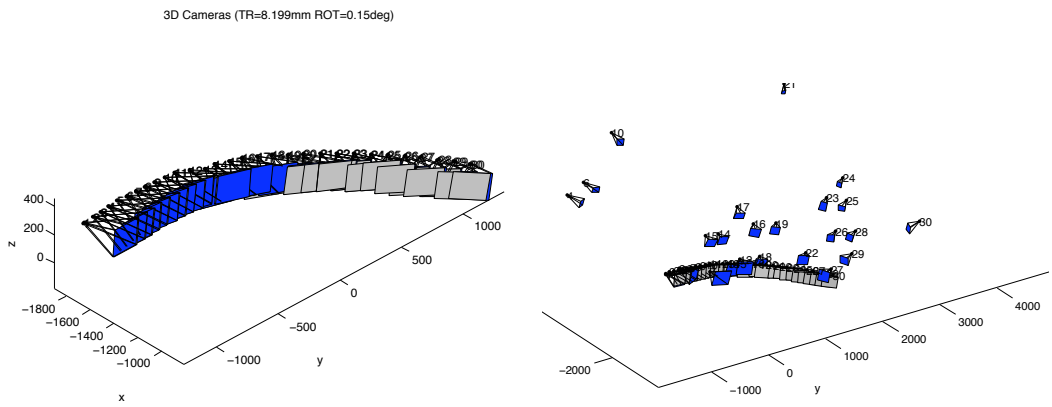


(b) *Corner Case*



(c) *Semi-Spherical Case*

Figure 4.9: Behavior of various robust and non robust algorithms when an increasing percentage of the input data are corrupted by high-amplitude noise, namely $\sigma = 0.2$ of the image size.



(a) Robust perspective factorization (b) Bundle Adjustment

Figure 4.10: Calibration results in the *Arc Case* for (a) the proposed method and for (b) bundle adjustment method, when 20% of the input data are corrupted with high-amplitude noise, namely $\sigma = 0.2$ of the image size. The 2-D reprojection error is of 0.71 pixels for (a) and 15.84 pixels for (b). The groundtruth is represented in gray.

4.9 Conclusions

In this chapter we described a robust factorization method based on data clustering and on the EM algorithm. First we recalled the classical maximum-likelihood approach within which all the observations are supposed to be independent and identically distributed. This amounts to classify all the observations in one cluster – inliers. Next we considered a mixture model within which the likelihood of the inlier class has a normal distribution and the likelihood of the outlier class has a uniform distribution. This naturally leads to ML with missing variables which is solved in practice via the Expectation-Maximization algorithm. We formally derived the latter in the specific case of 3-D reconstruction and of a Gaussian/uniform mixture; This allowed us to rely on EM's convergence properties.

Moreover, we devised two shape and motion algorithms: (i) affine factorization with EM and (ii) robust perspective factorization, the former residing in the inner loop of the latter. These two algorithms are very general since they can accomodate with any affine factorization and with any iterative perspective factorization methods.

We performed extensive experiments with two types of data sets: multiple-camera calibration and 3-D reconstruction. We compared the calibration results of our algorithm with the results obtained using other methods such as the bundle adjustment technique and IRLS. It is interesting to notice that there is almost no

noticeable quantitative difference between our algorithm and a non-linear optimization method such as bundle adjustment. The 3-D reconstruction results obtained with a single camera and objects lying on a turntable are also very good. Whenever possible, we compared our results with ground-truth data, such as the external camera parameters provided by the Middlebury multi-view stereo data set. In order to further assess the 3-D reconstruction results, we used the output of the robust perspective factorization method, namely a cloud of 3-D points, as input of a mesh-based reconstruction technique.

Our Gaussian/uniform mixture model and its associated EM algorithm may well be viewed as a robust regression method in the spirit of M-estimators. We compared our method with IRLS using a truncated quadratic loss function. The results show that our method performs slightly better, although we believe that these results are only preliminary. A thorough comparison between outlier detection using mixture models on one side, and robust loss functions on the other side is a topic in its own right. In the future we plan to extend our method to deal the more difficult problem of multiple-body factorization.

Chapter 5

Mesh-Based Surface Evolution

In this chapter we explore a new mesh based solution to surface evolution. Numerous shape modeling applications use surface evolution in order to improve shape properties, such as appearance or accuracy. Both explicit and implicit representations can be considered for that purpose. However, explicit mesh representations, while allowing for accurate and compact surface modelling, suffer from the inherent problems of not being able to reliably deal with self-intersections and topology changes. As a consequence, a majority of methods choose implicit representations of surfaces, e.g. level set methods, that naturally overcome these issues. Nevertheless, these methods rely on space discretizations which introduce an unwanted precision-complexity trade-off. In this chapter we explore an explicit mesh-based evolution scheme that robustly handles topology changes and removes self intersections, therefore overcoming the traditional limitations of this type of approaches. To demonstrate its efficiency, we present results in the context of mesh morphing, as well as multi-view stereo 3-D reconstruction, modeling with real and noisy data obtained from images.

5.1 Introduction

In the process of modeling shapes, several applications resort to surface evolution to improve shape properties. For instance, shape surfaces are evolved so that their appearances are improved, as when smoothing shapes, or so that they best explain given observations as in image based modeling. The interest arises in several fields related to shape modeling: computer graphics, computer vision, medical imaging and visualization among others. Surface evolution is usually formulated as an optimization process that seeks for a surface with maximal energy with respect

to the desired properties. To this aim, surfaces can be represented in different ways, from implicit to explicit representations, and deformed in an iterative way during optimization. Polygonal meshes, while being one of the most widely used representation in shape modeling, turn out to be more problematic in evolution schemes. The main reasons for that is the inherent difficulty to handle topological changes and self-intersections that can occur during evolution. In this chapter, we introduce a novel method and an intuitive and efficient algorithm that solves this issue and allows for meshes to be deformed in a consistent way.

5.1.1 Related Works

Surface evolution has been widely studied over the last decades and two main directions have been followed with respect to the representation which is considered for surfaces: Eulerian and Lagragian methods.

Eulerian methods formulate the evolution problem as time variation over sampled spaces, most typically fixed grids. In such a formulation, the surface, also called the interface, is implicitly represented. One of the most successful methods in this category, the *level set method* [123, 125], represents the interface as the zero set of a function higher dimensional function. A typical function used is the signed distance function of the explicit surface, discretized over the volume. At each iteration the whole implicit function is moved. The explicit surface is recovered by finding the 0-level set of the implicit function [94]. Such an embedding within an implicit function allows to automatically handle topology changes, i.e. merges or splits. In addition, such methods allow for an easy computation of geometric properties, such as curvatures, and benefit from *viscosity solutions* - continuos weak solutions that admit provably consistent numerical schemes. These advantages explain the popularity of level set methods in computer vision [124] as well as in other fields, such as computational fluid dynamics [153] as well as computer animations of fluids [44]. Nevertheless, implicit representations exhibit limitations resulting from the grid discretization. In particular, the precision/complexity trade-off inherent to the grid has a significant impact on the computational efficiency and the proposed narrow-band solutions [2] or octree based implementations [95] only partially overcome this issue. In addition, as shown by Enright et al. [43], the level set method is strongly affected by mass loss, smearing of high curvature regions and by the inability to resolve very thin parts. Another objection is that level set methods are not appropriate for tracking surface properties, such as color or texture coordinates, which can be desirable in many image-based approaches (i.e. motion tracking). Thus, while providing a

solution for the intersection and topological issues within surfaces, implicit representations introduce a new set of issues for which careful solutions need to be crafted.

Lagrangian methods propose an approach where surfaces have explicit representations which are deformed over time. Such representations, meshes for instance, present numerous advantages, among which adaptive resolution and compact representation, as well as the ability to directly handle non-geometric properties over the surface, e.g. textures, without the necessity to reconstruct the interface. On the other hand, they raise two major issues when evolved over time, namely self-intersections and topology changes, which make them difficult to use in many practical scenarios. This is why typically non-intersections and fixed topology were explicitly enforced [126, 70]. As a consequence, and in spite of their advantages, they have often been neglected in favor of implicit representations which provide practical solutions to such issues. Nevertheless, solutions to these issues have been proposed. McInerney and Terzopoulos [109] introduced topology adaptive deformable curves and meshes, called T-snakes and T-surfaces. However, in solving the intersection problem, the authors use a spatial grid, thus imposing a fixed spatial resolution. In addition, only offsetting motions, i.e. inflating or deflating, are allowed. Another heuristic method was proposed by Lauchaud et al. [91] for mesh deformations. Merges and splits are performed in near boundary cases: when two surface boundaries are closer than a threshold and facing each other, an artificial merge is introduced; a similar procedure is applied for a split, when the two surface boundaries are back to back. Self-intersections are *avoided* in practice by imposing a fixed edge size. A similar method was also proposed by Duan et al. [41]. Alternatively, Pons et al. [130] proposed a mesh approach based on a restricted 3-D Delaunay triangulation. A deformed mesh is obtained by triangulating the moved vertices and assuming that the tetrahedra categorization, i.e. inside and outside, remains after the deformation. While being a robust and elegant solution, it nevertheless relies on the assumption that the input mesh is sufficiently dense such that the Delaunay triangulation will contain a good approximation of the surface.

Solid Modeling technologies are also worth mentioning, since they provide practical tools to represent and manipulate surface primitives. Methods in this domain fall into two categories: Constructive Solid Geometry (CSG) [47, 75] and Boundary Representation (B-Rep) [15, 22]. CSG methods represent shapes as a combination of elementary object shapes based on boolean operations. Alternatively, B-Rep methods adopt the more natural approach to represent the object boundary using vertices, edges and facets [5, 134]. Each representation has its ad-

vantages. While boolean operations on CSG objects are straightforward, a lot of effort is required to efficiently render CSG objects [133, 60]. On the other hand, it is much more difficult to implement boolean operations on boundary representations (multiresolution surfaces) [16, 118], whereas interactive rendering is trivial. While these methods propose solution for computing boolean operations of surfaces, to the best of our knowledge they do not deal with any extension needed to address self-intersecting meshes. Generally, the methods are mostly concerned with proper rendering of the resulting geometry than with the generation of correct manifolds in the case of self-intersections.

5.1.2 Contributions

We propose a novel mesh-based self-intersection removal algorithm with direct application to surface evolution, which has its origins in the works of Aftosmis *et al.* [3] and Jung *et al.* [80]. In [3] the interest is to recover the outside "wetted" surface obtained from multiple intersecting meshes. The output mesh is obtained by identifying facets, or part of facets, which are on the exterior. The work of [80] uses the same idea, applied in the context of mesh offseting. We extend these approaches to the more general situation of any mesh deformation, including topological changes (i.e. joins and splits). The main contribution is a method that is guaranteed to provide a mesh surface, i.e. a 2-D compact oriented manifold, given a mesh surface that has been deformed by an arbitrary deformation field defined over the mesh vertices. To this purpose, we identify all the degenerate cases that can occur when locating the outside facets. We introduce the topological split operation as a natural extension. Additionally, we introduce a novel seed triangle finding procedure required to locate an initial facet outside the surface. The robustness and flexibility of the algorithm is illustrated in the context of mesh morphing, showing several challenging examples. In addition, we have successfully applied our approach to surface reconstruction using multiple cameras. Recent methods in image based modeling [142] make use of surface evolution to obtain precise 3D models from multiple images. Our approach contributes in this field by providing an unconstrained mesh-based solution that allows for facets of all sizes as well as for topology changes, with the goal of increasing precision without sacrificing complexity.

The remainder of the chapter is organized as follows. Section 5.2 introduces TransforMesh, our proposed method that handles self-intersection and topology changes. Different aspects of the approach, such as topological changes, guarantees, numerical stability, time complexity and implementation notes are detailed in section 5.3. Section 5.4 introduces a generic mesh evolution algorithm that

uses TransforMesh. Results in the context of mesh morphing are detailed in Subsection 5.4.1, showing different challenging topological scenarios. Additionally, results in the context of multi view 3-D surface reconstruction are presented in Subsection 5.4.2, making comparisons with state-of-the-art approaches. Finally, we conclude in section 5.5.

5.2 The Method

As stated earlier, the main limitations preventing many applications from using meshes are self-intersections and topology changes, which can occur frequently during surface evolution. In this chapter, we show that such limitations can be overcome using a very intuitive geometrically-driven solution. In essence, the approach preserves the mesh consistency, i.e. 2-D manifoldness, by detecting self-intersections and considering the subset of the original mesh surface that is *outside* with respect to the mesh orientation. The input mesh should be a *compact oriented 2-D manifold*. More rigorously, the mesh should satisfy the following criteria: every edge belongs to exactly two flat faces; every vertex is surrounded by a single cycle of edges and faces. Non-orientable manifolds, such as the Klein bottle, depicted in Figure 5.1, cannot be considered, since following the normal information one could pass from the exterior of the surface onto the interior.

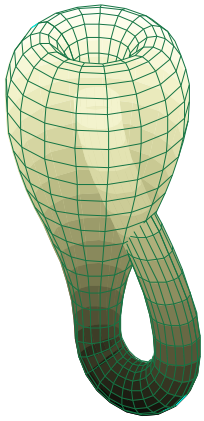


Figure 5.1: A Klein bottle - a typical example of a non-orientable 2-D manifold. By taking into account the normal orientation of a face and following it, one could pass from the exterior of the surface onto the interior.

Our proposed method, TransforMesh, has the great advantage of dealing with topological changes *naturally*, much in the same fashion as the Level-Set based

solutions, making it a viable robust alternative to surface evolutions with meshes. The requirements are that the input surface is a correct mesh surface, i.e. a 2-D compact oriented manifold, and that there exists an outside triangle without intersections for each connected component that needs to be recovered.

The idea behind the method is to find an initial seed triangle that is on the exterior, and then to propagate the *exterior* information by means of region growing. We call a *valid* triangle a triangle that is on the exterior without intersections and a *partial* triangle a triangle with intersections, that has one part the exterior. The algorithm, outlined in Figure 5.2, uses 3 queues for that purpose: one named $\mathcal{V}$, of valid triangles, i.e. triangles outside and without intersections; one named $\mathcal{P}$ of partially valid triangles, i.e. only part of the triangle is outside, and finally one named $\mathcal{G}$, where all the valid triangles will be stored until stitched together into a new mesh. The partial triangles are re-triangulated using a constrained 2-D Delaunay triangulation. An example is illustrated in Figure 5.3.

5.2.1 Self-intersections

The first step of the algorithm consists of identifying self-intersections, i.e. edges along which triangles of the mesh intersect.

This information will later on be needed in the computations, since it delimits the outside regions. We consider that triangles that share a simplex (edge or vertex) do not intersect. In the general situation, one would have to perform $O(n^2)$ checks to verify all triangle intersections, which can become quite expensive when the number of facets is large. In order to decrease the computational time, we use a bounding box test to determine which bounding boxes (of triangles) intersect, and only for those perform a triangle intersection test. We use the fast box intersection method implemented in [84] and described in [177]. The complexity of the method is $O(n\ log^d(n) + k)$ for the running time and $O(n)$ for the space occupied, where n is the number of triangles, d the dimension (3 in the current case), and k the output complexity, i.e., the number of pairwise intersections of the triangles.

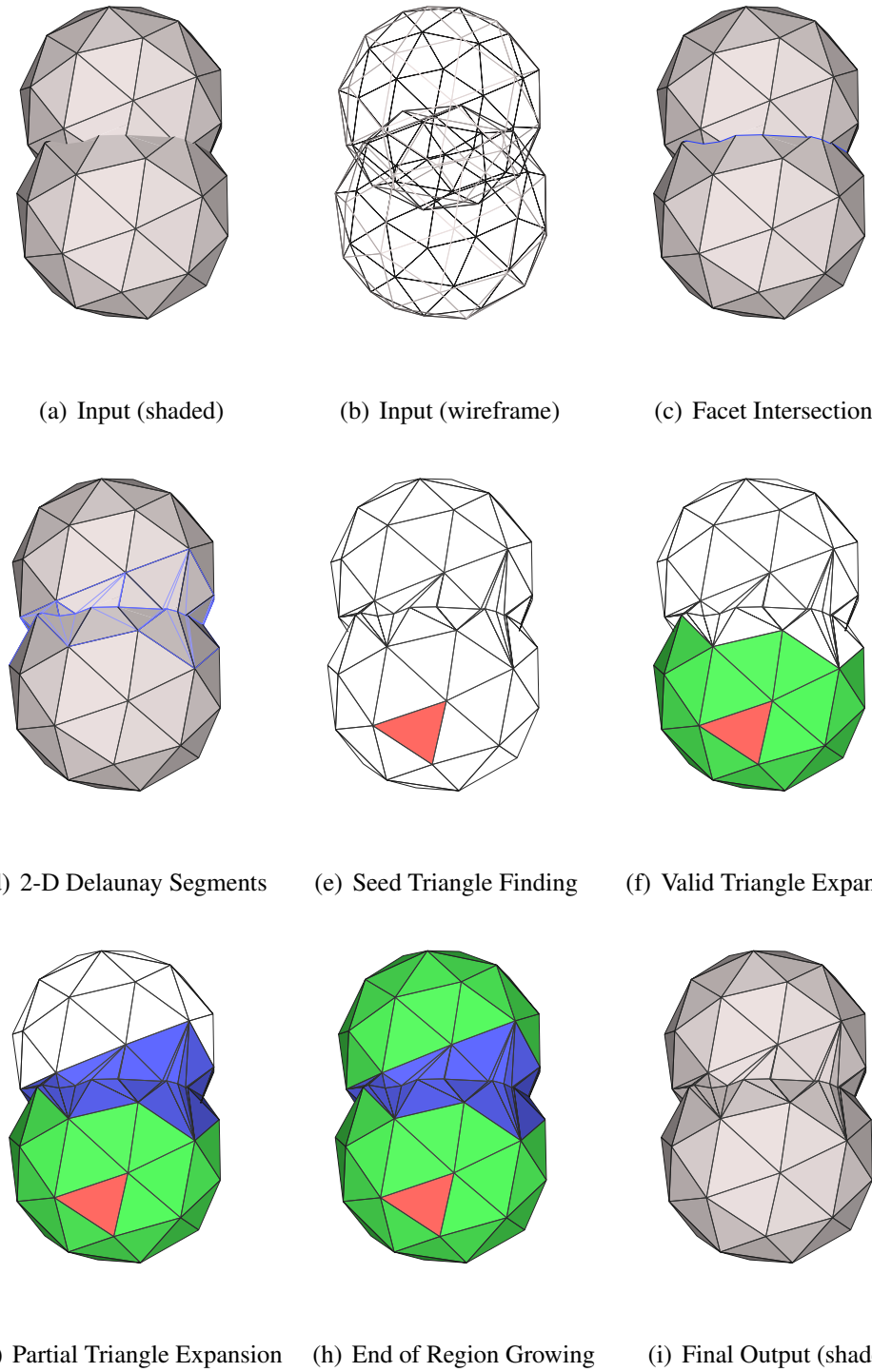
5.2.2 Valid region growing

The second step of the algorithm consists of identifying exterior triangles in the mesh. A valid region growing approach is used to propagate validity labels on triangles that composed the *outside* of the mesh. Alternatively, it can be viewed as a "painting" procedure, as it was described in [3]. Following this idea, we present

TransforMesh - Input: triangular mesh $\mathcal{I}$; Output: triangular mesh $\mathcal{O}$

1. Compute Self-Intersections (sec. 5.2.1) - compute all intersections bewteen triangles of $\mathcal{I}$
2. Valid Region Growing (sec. 5.2.2)
   - 2.1. Initialization (sec. 5.2.2) - mark all triangles as not visited
   - 2.2. Seed Triangle Finding (sec. 5.2.2) - find a valid outside triangle in $\mathcal{I}$ and add it to $\mathcal{V}$
   - 2.3 While $\mathcal{V} \neq \emptyset$
     - a. While $\mathcal{V} \neq \emptyset$ or $\mathcal{P} \neq \emptyset$
       - a.1. Valid Queue Processing (sec. 5.2.2) - $\forall t \in \mathcal{V}$, add t to $\mathcal{G}$, add valid unvisited neighbours to $\mathcal{V}$ and partial unvisited neighbours to $\mathcal{P}$ together with entrance edge.
       - a.2. Partial Queue Processing (sec. 5.2.2) - $\forall t \in \mathcal{P}$, perform a constrained 2-D Delaunay triangulation (intersection segments + edges), start from the entrance edge and select triangles to add to $\mathcal{G}$, stopping on constraint edges. Add corresponding triangles on the other side of the constraint edges to the appropriate queue, if not already visited.
     - b. Seed Triangle Finding (sec. 5.2.2) - find a valid outside triangle in $\mathcal{I}$ and add it to $\mathcal{V}$
   - 2.4 Triangle Stitching (sec. 5.2.3) - compute mesh $\mathcal{O}$ from $\mathcal{G}$

Figure 5.2: Algorithm layout. Each of the main steps of the algorithm corresponds to a section (marked in brackets) which details the necessary steps. In addition, Figure 5.3 outlines the region growing main steps.



(a) Input (shaded) (b) Input (wireframe) (c) Facet Intersections

(d) 2-D Delaunay Segments (e) Seed Triangle Finding (f) Valid Triangle Expansion

(g) Partial Triangle Expansion (h) End of Region Growing (i) Final Output (shaded)

Figure 5.3: An example of TransforMesh. The algorithm starts by computing all the self-intersections (c), followed by the valid region growing, which consists of first identifying a seed triangle (e) (marked in red), expanding on the neighbouring valid triangles (f) (marked in green) and selecting the correct subparts of the partially valid triangles (g) (marked in blue). The geometry of partial triangles is locally defined using a 2-D Delaunay triangulation (d). The procedure ends when all the valid and partially valid triangles have been visited (h).

here the sub-steps of the region-growing procedure. A *seed-triangle finding* algorithm is introduced, which is aimed at detecting start-up triangles without intersections that reside on the exterior. In the *valid queue processing* step this information is propagated by expanding on neighboring *valid* triangles (triangles with no intersections) until *partial* triangles (triangles with intersections) are reached. The *partial queue processing* steps details how to traverse the valid subparts of intersection triangles as well as how to ”cross” from one intersecting triangle to the other. The local subparts are triangulated using a constrained 2-D Delaunay triangulation. The underlying idea is to propagate the normal information from the seed triangles guided at the same time by the local geometry.

Initialization

Initially, all the triangles are marked as non-visited.

Seed-triangle Finding

A seed-triangle is defined as a non-visited triangle without intersections that resides on the exterior. This corresponds to Figure 5.3(e). The method determining if a triangle is on the exterior will be detailed below. In other words, a seed-triangle is a triangle that is guaranteed to be on the exterior. This triangle is crucial, since it constitutes the starting point for the valid region growing. If found, the triangle will be added to $\mathcal{V}$ and marked as valid; otherwise, the algorithm will jump to the next stage (section 5.2.3).

Proposed method: In order to verify if a triangle is on the exterior, we propose a new method, that extends the traditional ”point in polygon” test [66]. The original algorithm is formulated in 2-D and proposes that, given a point and a polygon, take a ray, originating at the given point, and count the number of times it intersects the polygon. If the number of times is even, then the point is on the exterior. A straight-forward extension to 3-D for our problem would be to consider a triangle, take a ray originating from the triangle centre along its normal and count the number of times it intersects other triangles. However, since in our scenario we deal with self-intersecting surfaces, the simple assumption that the line will pierce surfaces in and then immediately out does not hold. Such a counter-example is illustrated in Figure 5.4(a). Therefore, we propose to also take into account the orientation that each pierced facet makes with the incident ray. If there is an equal number of intersections between a ray, originating in the triangle centre along the normal direction, and facets oriented in each of the two directions, then the triangle is on the exterior (thus a valid seed triangle). A proof

follows from the fact that, as the original surface is closed, along any ray that pierces it there must be an equal number of intersections with both orientations. Then a squeeze or a twist of the surface will just add an even number of pairs of intersections with opposite directions along the line. We consider the *regularized* intersection, which ignores boundary cases, therefore considering that in the tangential case there is no intersection. In other words, given that the the oriented line intersects a triangle whose orientation is opposite, it must necessarily intersect later on another triangle whose orientation is alongside. The sign of the dot product between the two orientations is used in practice to determine the orientation. The proposed approach correctly finds the seed triangles even for surfaces of genus greater than zero, as it can be viewed in Figure 5.4(b), in the torus example.

In practice, in order to accelerate the test between a ray and a triangle, we compute the bounding box of the surface and its intersection with the ray, after which we sub-segment the obtained segment in smaller e_1 parts of length equal e_2 the average mesh edge size. This allows one to use the fast box-intersection test described earlier to compute the intersections between each sub-segment and other triangles. Therefore, it will take $O(e_1 \log(n))$ expected time to test if a triangle is a seed triangle, where n is the number of triangles of the surface.

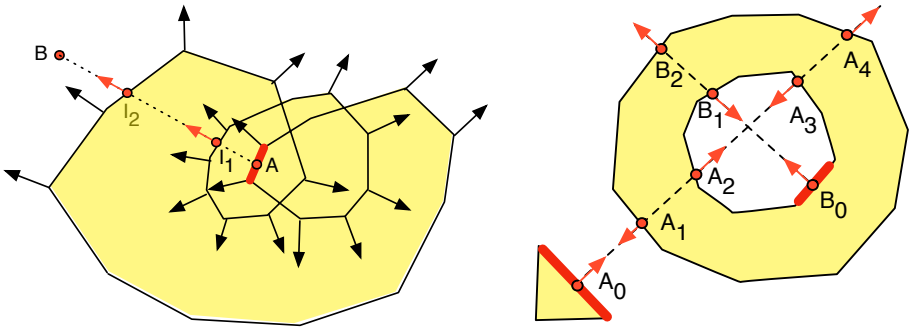
Extension: It is straightforward to extend the seed-triangle finding algorithm to take into consideration valid sub-parts of partial triangles. In order to keep the presentation clear, we will detail this extension in Subsection 5.3.3, after having introduced the algorithm.

Valid Queue Processing

Region growing is performed using the $\mathcal{V}$ queue, stopping on the intersections. Thus, while $\mathcal{V}$ is not empty, pop a triangle t from the queue, add it to $\mathcal{G}$ and for each neighbouring triangle $N(t)$ perform the following: if $N(t)$ is non-visited and has no intersections, then add it to $\mathcal{V}$; if $N(t)$ is non-visited and has intersections, then add it to $\mathcal{P}$ together with the entrance segment and direction, corresponding in this case to the oriented half-edge. (see Figure 5.3(f)).

Partially-Valid Queue Processing

Proper processing of regions containing intersections is ensured, with local geometry being generated. Thus, while $\mathcal{P}$ is not empty, pop a triangle t from the queue, together with the entrance half-edge f_t. Also, we have previously calculated all the intersection segments between this triangle and all the other triangles. Let $S_t = \{s_{ti}\}$ represent all the intersection segments between triangle t and the



(a) (b)

Figure 5.4: Seed Triangle Examples: (a) Example that illustrates why keeping track of triangle normal orientations is important when computing the number of intersections. Even though the number of the intersections between AB is even ($||\{I_1, I_2\}|| = 2$), the triangle, shown in red, is on the interior. (b) Example that illustrates different interesting seed triangle scenarios as well as the proposed solution. If a triangle is a seed triangle, the number of the projected normals of the intersection triangles (the red arrows) pointing inwards and outwards with respect to the original triangle normal should be equal. This is the case both for A_0 ($||\{A_2, A_4\}|| = ||\{A_1, A_3\}|| = 2$) and B_0 ($||\{B_1\}|| = ||\{B_2\}|| = 1$)

.

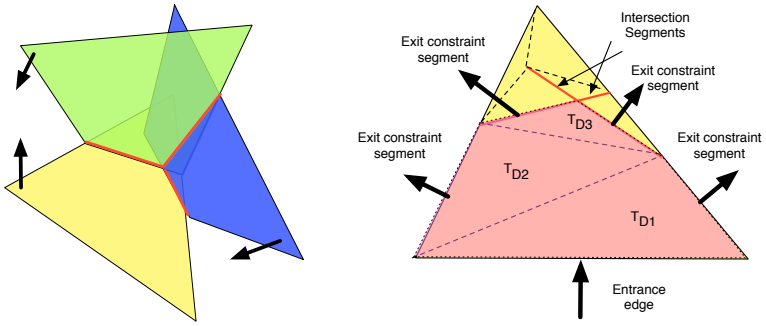
other triangles. In addition, let $H_t = \{h_{tj} | \text{for } j = 1..3\}$ represent the triangle half-edges. A constrained 2-D triangulation is being performed in the triangle plane, using [71], to ensure that all segments in both S_t and H_t appear in the new mesh structure and that propagation can be achieved in a consistent way. A fill-like traversal is performed from the entrance half-edge to adjacent triangles, stopping on constraint edges, as depicted in Figure 5.5. Choosing the correct side of continuation of the "fill" like region growing when crossing from a partially valid triangle to another is a crucial aspect in ensuring a natural handling of topological changes. The correct orientation is chosen such that, if the original normals are maintained, the two newly formed sub-triangles would preserve the water-tightness constraint of the manifold. This condition can also be casted as following: the normals of the two sub-triangles should be opposing each other when the two sub-triangles are "folded" on the common edge. A visual representation of the two cases is shown in Figure 5.6. The triangles on the other side of the exit constraint edges will be added to the appropriate $\mathcal{P}$ or $\mathcal{V}$ queues, based on whether they contain any intersections or not.

Note that it is possible to visit a partial triangle multiple times, depending on whether there are multiple isolated (non-connected) exterior components. The simplest example to image is a cross, formed out of two intersecting parallelipipeds. There will be partial triangles appearing on both sides. This is why in practice we mark the sub-parts of the triangulation already visited and allow for the re-visiting of partial triangles, if starting on a new sub-triangle.

5.2.3 Triangle Stitching

The region growing algorithm described previously will iterate until all the triangles have been selected.This corresponds to Figure 5.3(h). At this stage, what remains to be done is to stitch together the 3-D triangle soup ($\mathcal{G}$ queue) in order to obtain a valid mesh which is manifold. We adopt a method similar in spirit to [64, 144]. In most cases this is a straightforward operation, which consists of identifying the common vertices and edges between facets, followed by stitching. However, there are three special cases, in which performing a simple stitching will violate the mesh constraints and produce locally non-manifold structures. The special cases, shown in Figure 5.7, arise from performing stitching in places where the original structure should have been maintained. We adopt the naming convention from [64], calling them the singular vertex case, the singular edge case and the singular face case. All cases are easily identifiable by performing local operations.

Singular Vertex Case (Figure 5.7(a)): a vertex is shared by two or more dif-



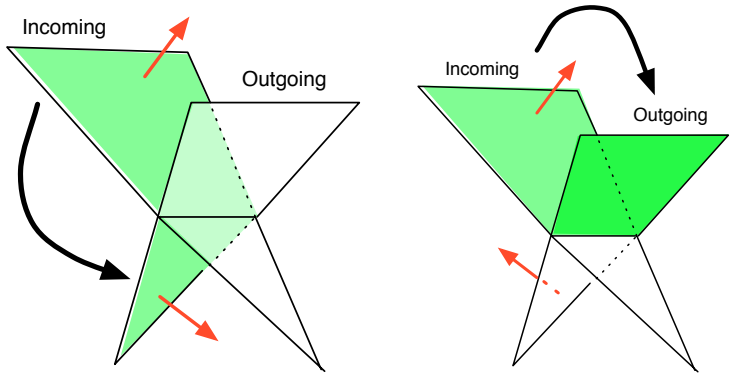
(a) Triangle Intersections (b) Partial Triangle Traversal

Figure 5.5: Partial Triangle Traversal. (a)The intersections with all other triangles are computed for each partial triangles. (b) The example shown here is for the bottom partial triangle presented in (a). The local geometry is re-defined using a constrainted 2-D Delaunay triangulation, where the constraints are the original triangle edges, the intersection segments and any of their intersections. The traversal follows the entrance edge and stops on constraint edges. To decide on which side to exit the constraints obtained from intersection segments, the partial triangle crossing procedures illustrated in Figure 5.6 is employed.

ferent regions. In this case, the manifold property stating that for each manifold point, there is a single neighbourhood, does not hold. Such situations occur in practice in rather convoluted topological situations, when a triangle gets inverted and intersects other facets. An example has been created to illustrate such a scenario and it is shown in Figure 5.8. The algorithm to detect these cases proceeds simply by checking that all facets incident to a vertex are within one neighbourhood. The steps are: mark all the facets incident to the vertex v as non-visited; starting from a facet of v, mark it visited and do the same with its non-visited neighbours that are also incident to v (neighbour as chosen based on half-edges); the process is repeated until all the neighbouring facets are processed; if by doing so we exhausted all the neighbouring facets, vertex v is non singular, otherwise it is singular, so a copy of it is created and added to all the remaining non-visited facets.

Singular Edge Case (Figure 5.7(b)): an edge is shared by two or more different regions, hence the manifold property does not hold. Such cases can be detected by counting the number of triangles that share an edge. If it is bigger than 2, then this is a singular edge case and two additional vertices and a new edge will be added to account for it. In practice, only the singular vertex cases appear.

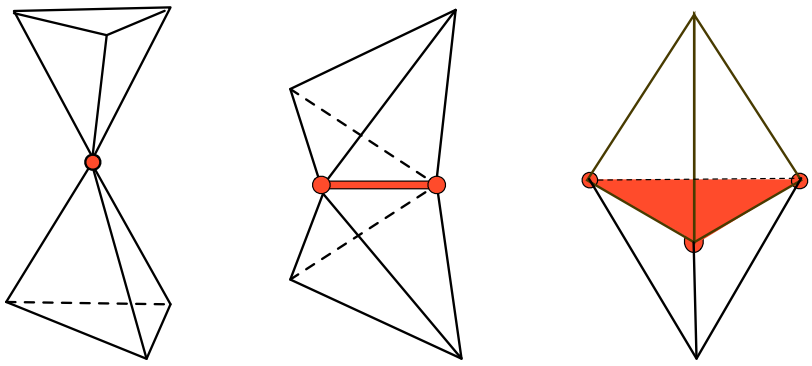
Singular Triangle Case (Figure 5.7(c)): a triangle is shared by two or more



(a) (b)

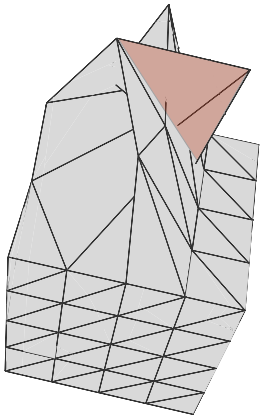
Figure 5.6: Partial Triangle Crossing Cases. Given the incoming triangle and its orientation, the outgoing triangle will be chosen such that the normals do not flip when attempting to cross them via the intersection segment.

different regions, hence the manifold property does not hold. Such situations never occur in practice. In any case, even if they did, they would be solved by the singular vertex processing phase, which would detect each of the 3 singular vertices that constitute the singular triangle.

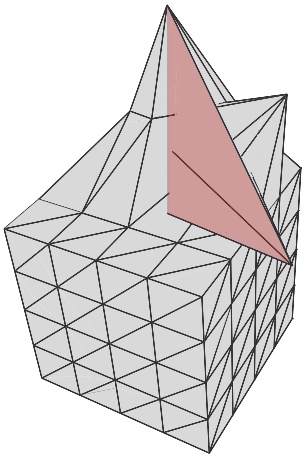


(a) Singular Vertex (b) Singular Edge (c) Singular Face

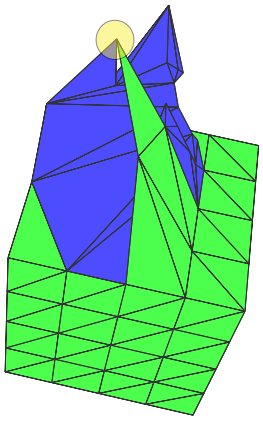
Figure 5.7: Special cases encountered while stitching a triangle soup.



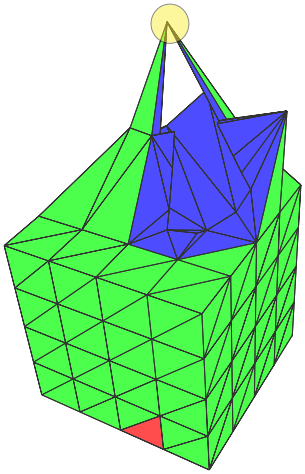
(a) Original Mesh (view 1)



(b) Original Mesh (view 2)



(c) Resulting Mesh (view 1)



(d) Resulting Mesh (view 2)

Figure 5.8: (a) (b) An example of how a singular vertex occurs in a typical self-intersection removal situation, due to an inverted triangle (marked in red). (c) (d) The singular vertex is circled in the resulting mesh. The coloring from the resulting mesh is corresponding to the color scheme used to exemplify TransforMesh.

5.3 Algorithm Features

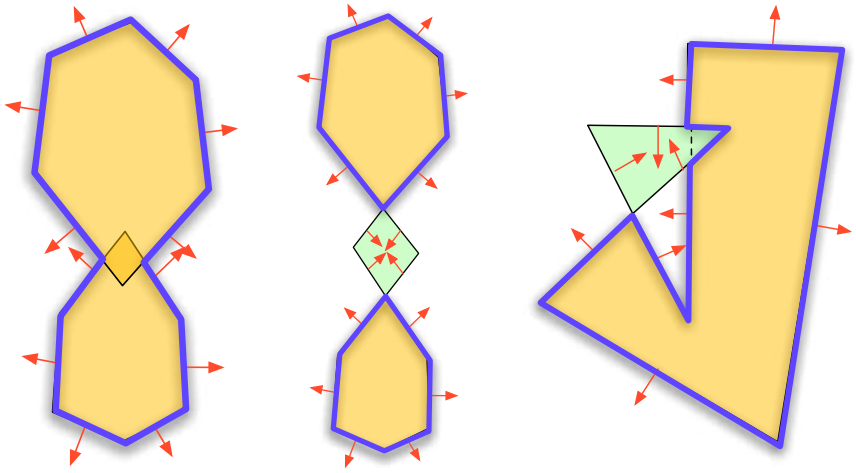
Having introduced the algorithm in the previous section, we will detail some of its important aspects, including the handling of topological changes, the guarantee to output a valid mesh, given a valid input mesh, numerical stability and time complexity.

5.3.1 Topological Changes

We consider compact surfaces. In the general case, the topological chances that can occur are: appear, disappear, join, split and genus change. The partial-triangle crossing technique described earlier in Section 5.2.2 and detailed in Figure 5.6 ensures a *natural* handling of the problematic topological changes, i.e. *splits* and *joins*, that plagued the mesh approaches until now. Representative cases are illustrated in Figure 5.9. The *join* case scenario, shown in Figure 5.9(a), coincides in spirit with the *union* boolean set operation $\cup^*$ and it is very easy to conceptualize. Less intuitive is the *split* operation, which will typically occur during a mesh evolution process, when certain parts will thin out up to the moment when some triangles from opposite sides will cross each other. Such a case is depicted in Figure 5.9(b), in a mesh morphing scenario, where the initial surface is one sphere and the destination is represented by two spheres. One additional example is presented in Figure 5.9(c), where an inverted geometry self-intersects the original mesh. The recovered surface is also a valid 2-D compact oriented manifold. A genus change can by tpyically viewed as the side-effect of a split / join operation. Additional examples will be presented in Section 5.4.1.

5.3.2 Guarantees

Given that the input mesh is a 2-D compact oriented manifold that has been deformed by a motion field and assuming exact computations (see section 5.3.4), TransforMesh will recover 2-D compact oriented manifold components. The number of components depends on the number of seed triangles detected. The algorithm will always *finish*, because it does not revisit already traversed subparts. In addition, it is guaranteed to always find the *exterior surface*, since it starts from a valid seed triangle (thus on the exterior) and it always rests that way, by propagating the normal information. The particular choice of the initial seed triangle does not influence the output of the method for that particular connected component. The computed output is *manifold* by construction, since it traverses a valid input manifold and accounts for the manifold violations (the degenerate cases). It



(a) Join Case (b) Split Case (c) Inside Carving Case

Figure 5.9: Different topological changes examples (2-D simplified view). The outline of the final surface obtained after self-intersection removal is outlined in bold blue.

is *compact*, since the original input surface has no borders and the algorithm does not build any (i.e. there is always a way outside a triangle intersection).

In addition, the 2-D manifold correctness is guaranteed by identifying and correcting all the possible 2-D manifold neighborhood violations when performing triangle stitching (singular vertex, singular edge and singular facet).

The algorithm preserves the geometry of the input mesh, with the exception of the self-intersection areas, where local triangulations redefine the geometry.

5.3.3 Seed-Triangle Finding Procedure Extension

The seed triangle finding procedure can be easily extended to take into account sub-triangles from partial triangles (triangles obtained from a 2-D constrained Delaunay triangulation on partial triangles and their intersection segments). This is possible thanks to the seed triangle finding method, described earlier, which will correctly identify the valid sub-triangles. If the seed-triangle is a valid sub-triangle, then it will be added to the partial queue $\mathcal{P}$. The rest of the overall algorithm would remain the same. In most of the typical scenarios this will not make any difference. Nevertheless, one could imagine some specifically designed examples (i.e. two opposite tetrahedrons centered within the same sphere), where the traditional algorithm will fail to find any connected component, due to the fact

that no start-up valid triangles can be found.

5.3.4 Numerical Stability

The numerical stability is critical, in order to be able to guarantee that the output is valid. It is ensured by using exact arithmetic predicates when computing intersections. Additionally, degenerate triangle intersection cases are explicitly handled. Special boundary cases between two intersection triangles are the following:

1. the intersection is a point;
2. the intersection is a 2-D polygon (when the two triangles are co-planar);
3. the intersection is a segment that lies on one of the original triangle edges.

The first and the second cases are boundary cases, therefore, by considering the *regularized* intersection operation $\cap^*$, we assume that there is no intersection between the triangles. In typical mesh evolution scenarios such situations do not occur, but it might be the case in CAD-designed datasets, where separate connected components are exactly arranged such that they overlap on a vertex/edge/facet. The third case, when the intersection is a segment lying on a original edge, turns out not to need a special treatment, due to the fact that the 2-D constrained Delaunay triangulation implementation of CGAL [18] properly deals with such situations, where constraints intersect.

One last possible degenerate scenario is when multiple triangle-triangle intersections take place at the same element. This situation can be disambiguated using the Simulation of Simplicity technique of virtual perturbations [42].

The rest of the numerical stability issues related to fixed numerical precision are handled by using exact geometrical predicates in CGAL. These are issues related to the fact that, due to fixed machine precision, newly computed intersection segments might not be strictly co-planar with the original triangles. Similarly, degenerate triangles could be generated by the 2-D Delaunay triangulation.

In practice, we also eliminate triangles having area close to zero, should they occur. In order to remove those triangles, two operations are performed: edge collapse and edge flip.

5.3.5 Time Complexity

The time complexity of the algorithm depends on the number and relative sizes of facets and it is of $O(n \log(n))$ for the generic case. Most of the time will be spent

computing the intersections. Even if there are no intersections, the same relative time will be spent, since each triangle needs to be checked and each check will take an amortized time of $O(\log n)$. In practice, more than 80% of the running time is spent computing the self-intersections. Typically, the running time for performing the self-intersections test is under 1 second for a mesh with $50,000$ facets on a 2.6 GHz Intel Core2Duo (no multi-threading), where exact arithmetic is used for triangle intersections and the self-intersections are in the range of 100.

5.3.6 Comparison with a static algorithm

Alternatively, one could use the seed triangle finding procedure, described in Section 5.2.2, in order to devise another static algorithm, which will test all the existing valid triangles and partial subtriangles obtained from local Delaunay triangulations. The meothod will only choose the triangles that pass the seed triangle test, thus residing on the exterior, after which it will proceed to the final triangle stitching step.

However, this static algorithm would take a considerable longer time, since it requires the same initial time $O(n \log n)$ to compute all the triangle intersections and local Delaunay triangulations. It will require an additional $n_1 e_2 \log n$, where n_1 is the number of $n_1 > n$ is the total number of valid triangles and partial subtriangles obtained from Delaunay triangulations, and e_2 is the average number of segments that a ray was chopped into, when performing the seed triangle test, as described in Section 5.2.2. On the positive side, this static algorithm is much simpler to implement than the proposed one.

5.3.7 Extension to Open Surfaces

The currently proposed method extends without any modifications to open surfaces (i.e. triangular meshes with holes), since the algorithm relies on local normal information in order to infer the interior and exterior. However, the seed-triangle finding procedure can potentially miss correct triangles or choose incorrect ones in specially designed cases. The seed-triangle finding procedure assumes closed surfaces when performing the counting of the number of intersections. An alternative "safer" seed-triangle procedure method can be employed, choosing only the triangles whose normal extension does not intersect any other triangles. Nevertheless, in practice, in typical scenarios, the algorithm can be used as presented, as it will be shown in Section 5.4.1.

5.3.8 Implementation Notes

In our implementation we have made use of CGAL (Computational Geometry Algorithms Library) C++ library [18], which provides *guaranteed* implementations for various algorithms. We have used the following CGAL modules: n-dimentional fast box intersections, 2-D constrained Delaunay triangulation, triangular meshes and support for exact arithmetic kernels.

5.4 Applications

TransforMesh can be easily incorporated within a **generic mesh evolution** paradigm, as illustrated in Figure 5.10. Within each evolution iteration, there are four steps. Firstly, a velocity vector field $\vec{\mathcal{F}}$ is computed for each vertex of the mesh $\mathcal{M}$. This step is application specific. Secondly, the mesh is evolved (moved) using the computed velocity vector field $\vec{\mathcal{F}}$ and a small time step t, thresholded by a maximum movement $\alpha \cdot e_{avg}(v)$, where α is a user-set threshold (typically between 0.1-0.3) and $e_{avg}(v)$ represents the local average edge length for a vertex v. Thirdly, TransforMesh is invoked in order to clean the potential self-intersections and topological problems introduced by the second step. The fourth step involves mesh optimization, with the goal of ensuring good mesh properties. Ideally, a mesh should consist of triangles as close to equilateral as possible, which allows for better computations of local mesh properties, i.e. curvature, normal. To this purpose, a number of sub-steps are being performed: adaptive remeshing, vertex valence optimization and Laplacian smoothing. The adaptive remeshing steps ensures that all edges are within an safety zone interval (e_1, e_2), user-defined. This prevents edges from reaching close to zero sizes. In practice, this is obtained via edge collapse or edge swap operations. The accumulation of edge collapses is prevented, as suggested in [87], by collapsing towards the vertex with higher valence. Vertex valence optimization step performs edge swaps in an attempt to ensure an overall vertex valence of 6 [87]. Vertex valence is defined as the number of triangles shared by a vertex. The ideal vertex valence of 6 is desirable because, assuming that the manifold is generally locally planar, it is equivalent to obtaining 60° for each of the sharing triangle angles, thus optimizing for equilateral triangles. The Laplacian smoothing is attained by computing the discrete mesh Laplacian [36, 113] (i.e. the Laplace-Bertrami operator) Δv for each vertex v of the mesh. Furthermore, the mesh is smoothed using $v \rightarrow v - \beta \Delta v$. These four main steps are repeated until the mesh has reached the desired final state, also application specific.

We will present below two examples, one for Mesh Morphing in Section 5.4.1

and the other for Multi-View 3-D Reconstruction in Section 5.4.2. In both cases, the application specific information is detailed in order to compute the vector fields $\vec{\mathcal{F}}_{morphing}$ and $\vec{\mathcal{F}}_{reconstruction}$, which plug directly within the generic mesh evolution framework presented in Figure 5.10.

Generic Mesh Evolution using TransforMesh

While Not Finished

1. **Compute Velocity Vector Field** $\vec{\mathcal{F}}$ of velocities for each vertex of the mesh $\mathcal{M}$, using application specific information.
2. **Evolve Mesh** $\mathcal{M}$ using the vector field $\vec{\mathcal{F}}$ and a small time-step t, thresholded by a maximum $\alpha \cdot e_{avg}(v)$, where α is a user-set threshold (typically set between 0.1-0.3) and $e_{avg}(v)$ is the local average edge length for a vertex v.
3. **Invoke TransforMesh** on $\mathcal{M}$ in order to clean self-intersections and topological problems
4. **Mesh Optimization**
   a) *Adaptive Remeshing*: ensures that all edges are within a safety zone interval, i.e. $\forall e \in \mathcal{M}, e \in [e_1, e_2]$, by performing edge swaps or edge collapses.
   b) *Vertex Valence Optimization*: perform edge swaps such that each vertex will be shared by 6 triangles [87].
   c) Perform *Mesh Laplacian Smoothing*: each vertex v is updated by $v \rightarrow v - \beta\Delta v$, where Δv represents the discrete mesh Laplacian [36, 113].

Figure 5.10: Generic Mesh Evolution Algorithm using TransforMesh.

5.4.1 Mesh Morphing

A first and straight-forward application of our algorithm is surface morphing, that is starting from a source surface S_A and evolving it towards a destination surface S_B. This will allow us to test thoroughly various cases of topology changes. Surface morphing has been widely described in the literature. We will adopt the

method proposed by Breen et al. [24]. We will summarize the reasoning that leads the surface evolution equation.

Methodology

A metric that quantifies how much two surfaces overlap is defined (source surface S_A and destination surface S_B). A natural choice of such a metric is the signed distance function γ_B of the destination mesh S_B', defined as in the level set literature as being negative inside the shape S_B, zero on the surface, and positive on the exterior. By considering the volume integral $\mathcal{M}_{S_B}(S_A)$ over any surface S_A with respect to γ_B (thus S_B), one can see that it will achieve the maximum when the two surfaces overlap. By taking the first variation of the metric $\mathcal{M}_{S_B}(S_A)$ with respect to the surface S_A and a small displacement field and differentiating with respect to the vector field, one obtains the following evolution equation using a hill climbing strategy for each vertex x along its normal $\mathbf{N}(x)$:

$$\vec{\mathcal{F}}_{morphing} = \frac{\partial S}{\partial t} = -\gamma_B(x)\mathbf{N}(x) \tag{5.1}$$

The evolution strategy described above will converge to a local minimum. Given the surface of departure S_A and the destination surface S_B, S_A will correctly find all the connected components of S_B that are included in the original surface S_A. That is to say that, if S_A represents a surface outside the destination surface S_B, S_A will converge to an empty surface. We keep this result in mind when choosing the initial surface S_A.

Complexity Issues and Mesh Discretisation

In the general case, in order to calculate an exact distance function γ_B, one would have to consider the projection of a query point to the planes defined by each facet of the mesh (representing the surface S_B) and to see if the projection lies within the interior of the facet, keeping the closest distance. This lookup will take $O(N_F)$, where N_F represents the number of facets. This is a fairly expensive computation, which will have to be performed at each iteration throughout the evolution for every vertex.

One approximate solution used in the level-set literature [123] is to obtain a space discretized signed distance function using a 3-D grid, by computing the intersections of the mesh with the grid (thus finding the 0 level set) and performing fast marching [79] in order to fill out the 3-D grid. This solution, while benefitting

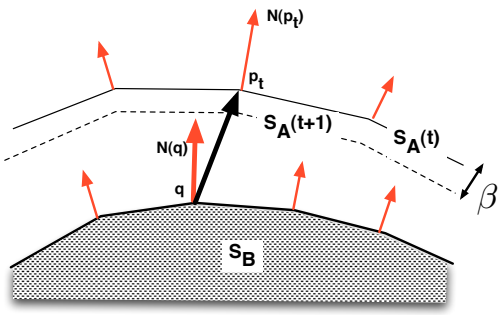


Figure 5.11: Mesh Morphing evolution step. The surface S_A evolves from time t to time $t+1$ towards S_B. For point p_t, the closest point is q. The evolution equation for point p_t is given by $(p_t - q) \cdot N(p_t)$

from a $O(1)$ lookup cost, it uses memory proportional to the space discretization and it takes $O(G)$ to initialize the distance function, where G represents the number of grid elements. This approximation is valid up to grid cell size.

Alternatively, as an in-between tradeoff, we propose a approximation/heuristic using the distance to the closest vertex point. This speeds up computation drastically, since, if proper search structures are being used, the search time for the nearest neighbour is $O(log(N_V))$, where N_V to represent the number of vertices. There is an initial overhead of $O(N_V log(N_V))$ of building the search tree. In practice, we have used the implementation of [72] available in CGAL. Please note that if the target surface S_B contains a good enough mesh resolution, this approximation is very close to the true signed distance function.

In the case of sufficient sampling, the current approximation will return a vertex belonging to the closest triangle where the true projection would be. Thus, the error bound is the distance between the vertex and the projection. In practice, however, we do not use the actual distance, but its sign, in order to establish the direction of the evolution. This makes the current approximation fit for our purpose. Alternatively, one could easily verify all the incident triangles to the closest vertex to establish the true distance function, if the application requires it, keeping in mind that the sufficient sampling condition still applies. Alternatively, in order to obtain accurate results, a full distance field implementation can be considered. A comprehensive review of such methods can be found in [79].

The proposed solution has the great advantage of being able to be applied in the current formulation, not only to meshes, but also to oriented 3-D points. The source surface to be evolved is still a mesh, but the destination surface can be represented by oriented 3-D points. If orientation information is not available,

it can be estimated from neighboring points using principal component analysis [73]. Alternatively, in the context of multi-view stereo, it can be obtained via a minimization scheme [54].

Results

In Table 5.1 we present results obtained with four test cases, entitled "Genus 3", "Toruses", "Knots In" and "Knots Out". As it can be observed, the algorithm successfully deals with topology joins and splits as well as handling multiple connected components. The average computation time per iteration on a 2.6 GHZ Intel Core2Duo processor varies between 0.2 to 1.6 seconds, depending on the number of facets and on the number of intersections. More detailed statistics are presented in Table 5.2.

In Table 5.3 we present additional evolution results obtained when using open surfaces, in order to show that our method deals gracefully with surfaces with holes, without explicitly modeling it.

In terms of parameters setting with respect to the generalized mesh evolution framework depicted in Figure 5.10 within which we casted the current mesh morphing algorithm, we considered $t = 1$ for the timestep, $\alpha = 0.2$ the average edge size e_{avg} for maximum movement amplitude and $\beta = 0.1$ for the smoothing term. Additionally, the original meshes had a constant mesh resolution. Hence, we set the edge thresholds to $e_1 = 0.7 \cdot e_{avg}$ and $e_2 = 1.5 \cdot e_{avg}$.

5.4.2 Multiple Camera 3-D Reconstruction

Our original motivation in developing a mesh self-intersection removal algorithm was to perform mesh evolutions, in particular when recovering 3-D shapes from multiple calibrated images. As stated earlier, few efforts have been put in mesh-based solutions for the 3-D surface reconstruction problem, mostly due to the topological issues raised by mesh evolutions. However, meshes allow one to focus on the region of interest in space, namely the shape's surface and, as a result, lower the complexities and lead to better precisions with respect to volumetric approaches. In this section we present the application of TransforMesh to the surface reconstruction problem. Often such a problem is casted as an energy minimization over a surface. We start from exact visual hull reconstructions, obtained with [49] using silhouette information and further improve the mesh using photometric constraints by means of an energy functional described in [131]. In their original implementation, [131] used a level set formulation. Our goal is to maintain the energy functional, while replacing the surface evolution method.

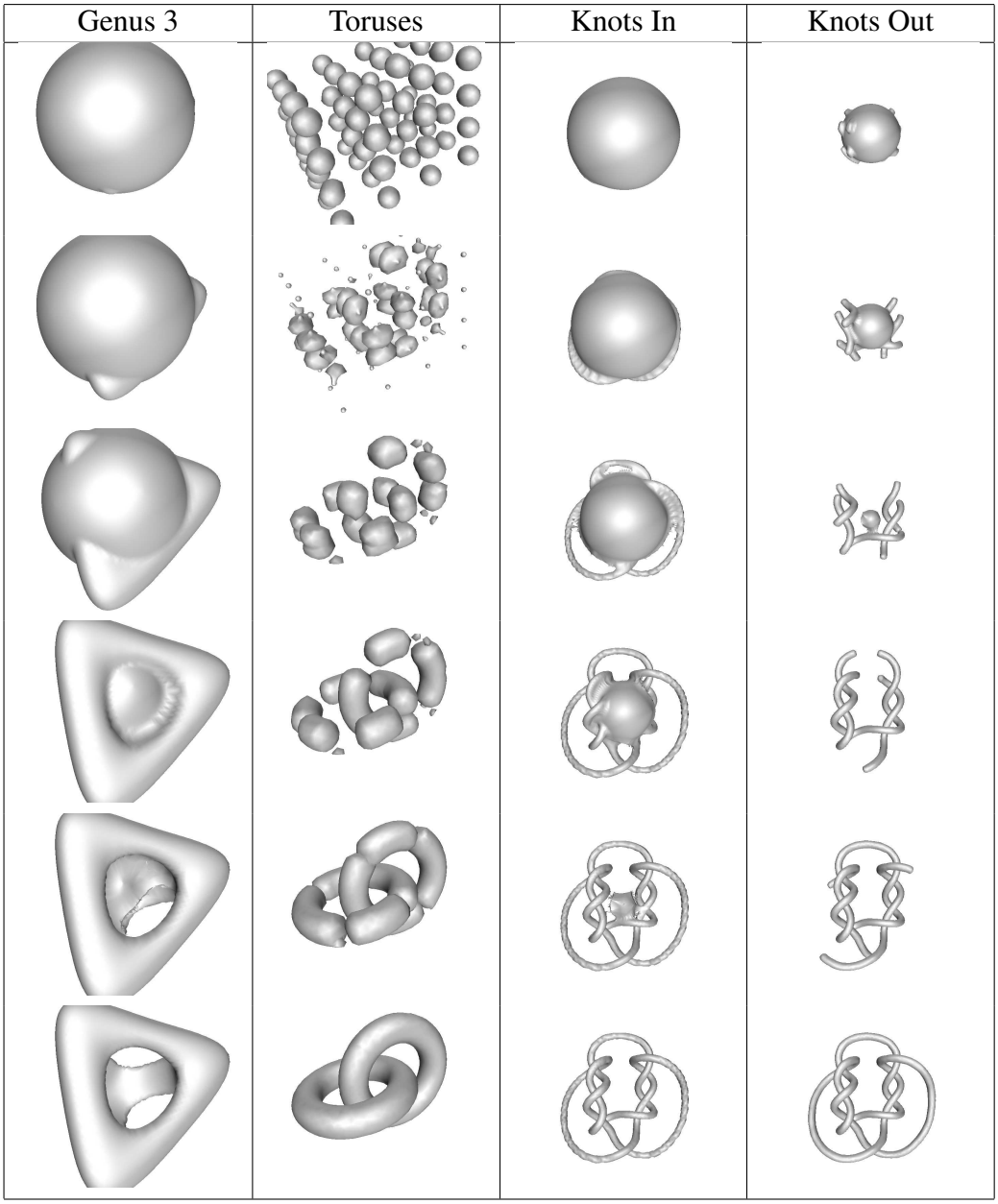


Table 5.1: Mesh Morphing Examples with Closed Surfaces. Different steps during for various test cases. Each test case is presented in a column. The first row represents the first iteration, whereas the last row represents the last iteration.

Dataset	Iterations	Avg. # Facets	Avg. # Intersections	Avg. Time (SIR)	Avg. Time (total)
Genus 3	54	4764.14	33.88	0.65 sec	1.42 sec
Toruses	37	6296.33	22.67	0.81 sec	1.78 sec
Knots In	119	13244.25	101.52	1.63 sec	3.58 sec
Knots Out	430	3873.11	4.86	0.18 sec	0.89 sec

Table 5.2: Mesh Morphing Statistics for different datasets. The running time is recorded on a 2.6 GHz Intel Core2Duo processor. SIR in the table header denominates self-intersection removal (TransforMesh).

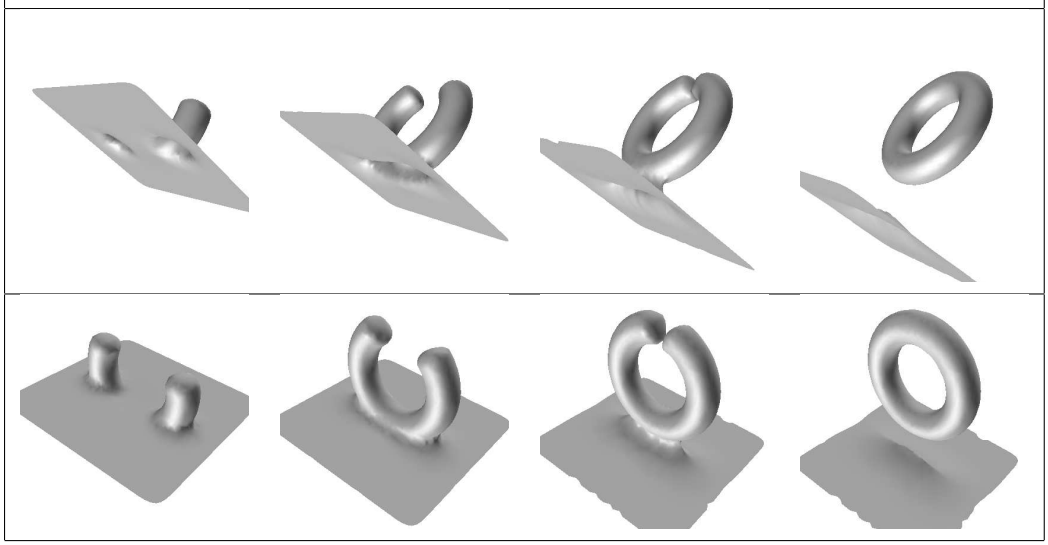
Torus


Table 5.3: Mesh Morphing Example with an Open Surface. The evolution starts from an open plane surface, going towards a torus. Each row represent the evolution at different steps, shown from a different point of view.

Methodology

The initial mesh surface corresponds to an extended bounding box obtained using image silhouettes and a geometric approach that involves cone intersections in 3-D (see [49] for details). Such a mesh is only a coarse approximation of the observed surface. One main limitation of visual hull approaches is that they do not recover concave regions. The initial surface can be improved by considering photometric information in the images. The principle is that with a correct geometry (and under the Lambertian light assumption) the mesh should be photo-consistent, i.e. its projections in the images should have similar photometric information [9].

We adopt the formalism proposed by Jean-Philippe Pons et al. [131] as well as the gradient computation code kindly made available to us by the authors. The photometric constraints are casted as an energy minimization problem using similarity measure between pairs of cameras that are close to each other. We denote by $S \subset \mathbb{R}^3$ the 3-D surface. Let $I_i : \Omega_{\mathrm{i}} \subset \mathbb{R}^2 \rightarrow \mathbb{R}^d$ be the image captured by camera i (d=1 for grayscale and d=3 for color images). The perspective projection of camera i is represented as $\Pi_{\mathrm{i}} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Since the method uses visibility, consider S_i as part of surface S visible in image i. In addition, the back-projection of image from camera i onto the surface is represented as $\Pi_{\mathrm{i}}^{-1} : \Pi_{\mathrm{i}}(\mathrm{S}) \rightarrow \mathrm{S_i}$.

Armed with the above notation, one can compute a similarity measure M_{ij} of the surface S as the similarity measure between image I_i and the reprojection of image I_j into the other camera i via the surface S. Summing across all the candidate stereo pairs, one can write:

$$\mathcal{M}(S) = \sum_i \sum_{j \neq i} \mathcal{M}_{ij}(S) \tag{5.2}$$

$$\mathcal{M}_{ij}(S) = M|_{\Omega_i \cap \Pi_i(S_j)}\left(I_i, I_i \circ \Pi_i \circ \Pi_{j,S}^{-1}\right) \tag{5.3}$$

Finally, the surface evolution equation at a vertex point x is given by:

$$\vec{\mathcal{F}}_{reconstruction} = \frac{\partial S}{\partial t} = E_{img}(x)\mathbf{N}(x) \tag{5.4}$$

where $\mathbf{N}$ represents the surface normal and E_{img} is a photoconsistency term that is a summation across pairs of cameras which depends upon derivatives of the similarity measure $\mathcal{M}$, of the images I, of the projection matrices Π and on the distance x_z (see [131] for more details). We have used the normalized cross correlation as the similarity measure, with a support of 5 pixels.

In the original paper [131], the surface evolution equation was implemented within the Level-Set framework. We adapt it to meshes using the TransforMesh algorithm. The original solution performs surface evolution using a coarse to fine approach in order to escape local minima. Traditionally, in Level-Set approaches, the implicit function that embeds the surface S is discretized evenly on a 3-D grid. As a side-effect, all the facets of recovered surface are of approximately equal triangle size. In contrast, mesh based approaches do not impose such a constraint and allow facets of all sizes on the evolving surface. This is particularly useful when starting from visual hulls, for which the initial mesh contains triangles of all dimensions. In addition, the dimension of visual facets appears to be a relevant information since regions where the visual reconstruction is less accurate, i.e. concave regions on the observed surface, are described by bigger facets on the visual hull. Thus, we adopt a coarse to fine approach in which bigger triangles are moved until they stabilize, after which the whole process is repeated at a finer scale.

Results

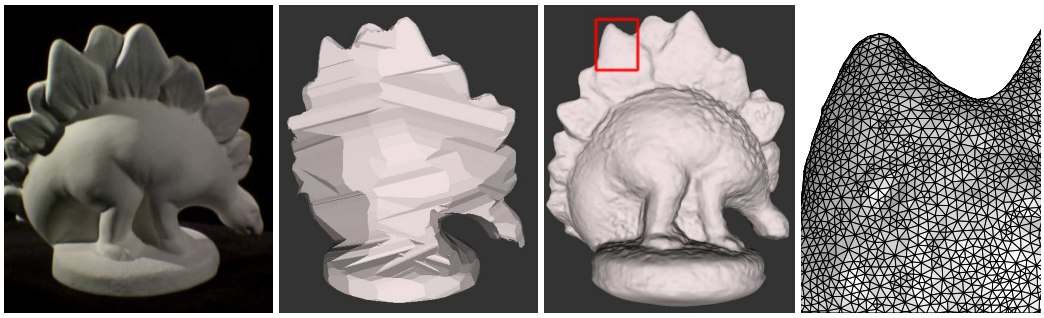
We have tested the mesh evolution algorithm with the datasets provided by the Multi-View Stereo Evalutation site [142] (http://vision.middlebury.edu/mview/) and our results are comparable with state-of-the-art, attaining sub-milimeter accuracy. Detailed results are extracted from the website and are presented in Table 5.4. We have also included results by Furukawa et al. [54], Pons et al. [131] and Hernandez et al. [70], considered to be the state of the art. The differences between all methods are very small, ranging between $0.01mm$ to $0.3mm$. Some of our reconstruction results are shown in Figure 5.12.

In terms of parameters setting with respect to the generalized mesh evolution framework depicted in Figure 5.10 within which we casted the current multi-view stereo reconstruction algorithm, we considered $t = 0.001$ for the timestep, $\alpha = 0.1$ the average edge size e_{avg} for maximum movement amplitude and $\beta = 0.1$ for the smoothing term. The meshes had an adaptive mesh resolution. As mentioned earlier, we ran the algorithm at different scales, starting from scale s_{max} to $s_{min} = 1$ in $\lambda = \sqrt{2}$ decrements. For each scale s_i, the input images and camera matrices are downscaled accordingly. The appropriate edge size interval is set to $e_1 = edgeSize(1, 1)$ $e_{2i} = edgeSize(5, i)$, where $edgeSize(p_1, p_2)$ is a function that computes the desired edge size such that it has p_1 pixels in images, where the images have been downscaled by p_2. The startup scale s_{max} is computed such that the larger edges of the startup mesh measure 5 pixels when projected into the images at scale s_{max}. When the finer scale is reached, new iterations are run by decreasing e_2 from $edgeSize(5, 1)$ to $edgeSize(2, 1)$ in $\lambda = \sqrt{2}$ decrements.

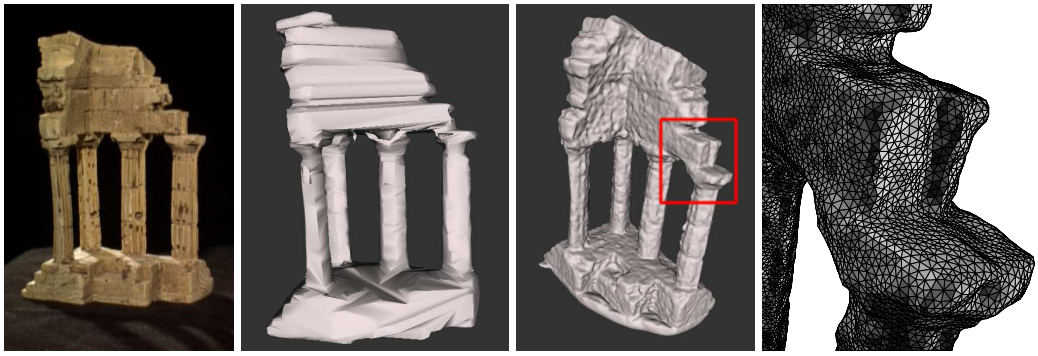
Dataset / Paper	Temple Ring		Temple Sparse Ring		Dino Ring		Dino Sparse Ring	
	Acc.	Compl.	Acc.	Compl.	Acc.	Compl.	Acc.	Compl.
Pons et al. [131]	0.60mm	**99.5%**	0.90mm	95.4%	0.55mm	99.0%	0.71mm	97.7%
Furukawa et al. [54]	0.55mm	99.1%	**0.62mm**	**99.2%**	**0.33mm**	**99.6%**	**0.42mm**	**99.2%**
Hernandez et al. [70]	**0.52mm**	**99.5%**	0.75mm	95.3%	0.45mm	97.9%	0.60mm	98.52%
Our results	0.55mm	99.2%	0.78mm	95.8%	0.42mm	98.6%	0.45mm	**99.2%**

Table 5.4: 3-D Rec. Results. Accuracy: the distance d in mm that brings 90% of the result R within the ground-truth surface G. Completeness: the percentage of G that lies within 1.25mm of R.

The algorithm converges to a good solution without the presence of a silhouette term in the evolution equation. In a typical evolution scenario, there are some self-intersections at the beginning, but, as the algorithm converges, intersections rarely occur. Additionally, in the temple case, we performed a test where we have started from one big sphere as the startup condition, in order to check whether the topological split operation of TransforMesh performs properly. Proper converges was obtained.



(a) Dino - Input (b) Dino - Start (c) Dino - Results (d) Dino - Closeup



(e) Temple - Input (f) Temple - Start (g) Temple - Results (h) Temple - Closeup

Figure 5.12: Reconstruction results obtained in the temple and in the dino case.

We present some additional results obtained for the Dance-2 sequence pub-

licly available from the Multiple-Camera/Multiple-Video Database of the PERCEPTION group [1] in Figure 5.13.

Figure 5.13: Results for the Dance-2 sequence from INRIA. We present results for frames 500 to frame 575 in jumps of 5 frames.

[1] https://charibdis.inrialpes.fr/html/index.php

The *Leuven* [2] sequence consists of 7 high resolution images 3000×2000 images of Leuven's City Hall. The final reconstruction contains a mesh with over 1.5 million triangles. Results are presented in Figure 5.14. An adaptive remeshing procedure has been applied in order not to further refine the invisible triangles.

The *Staty* sequence consists of 32 images of resolution 720×576. The reconstruction results are presented in Figure 5.15. We have also presented a mean curvature view in order emphasize some of the high detail structures reconstructed, such as the cobblestone.

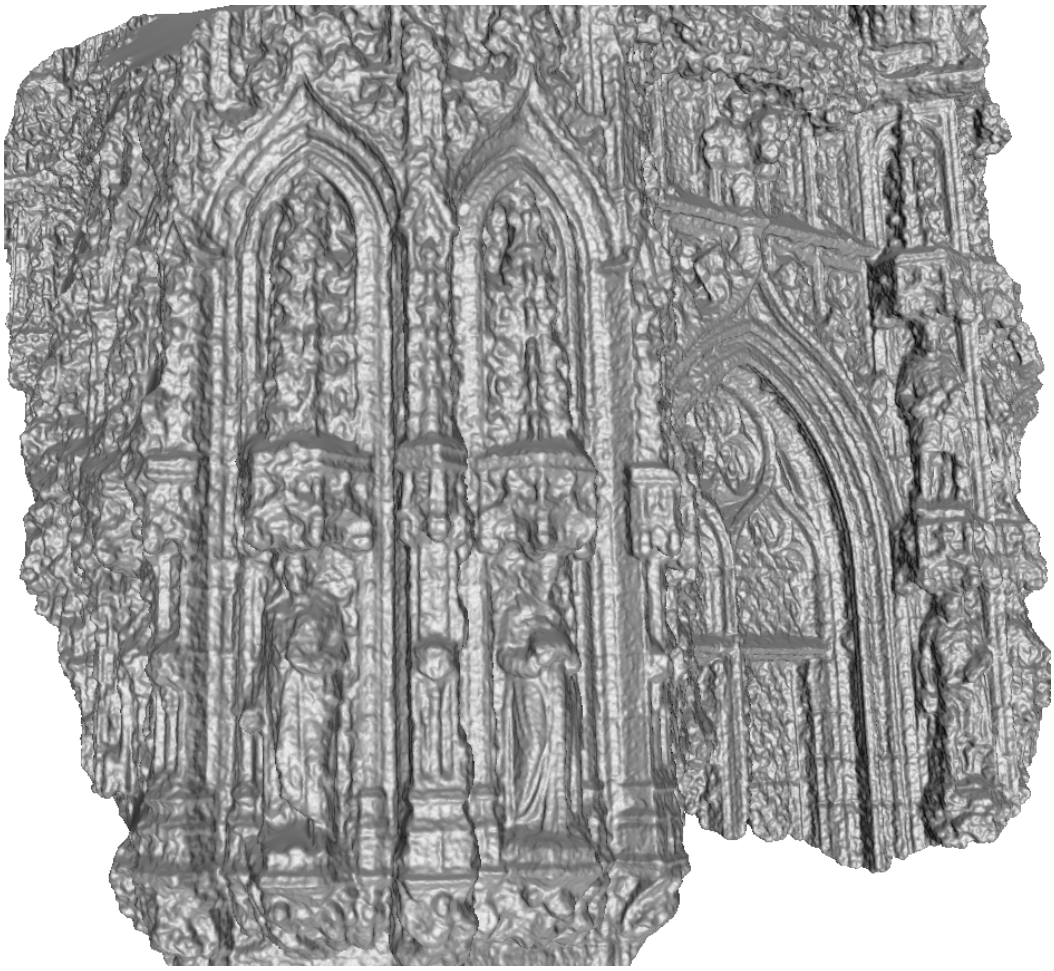
In order to make the link with the previous chapter, we used the 3-D reconstructed points obtained using the robust perspective factorization method presented in chapter 4 and combined it with the PowerCrust algorithm [12] in order to obtain an initial mesh surface, which was later on evolved using the currently presented scheme. The results are shown on Figure 5.16.

5.5 Conclusion

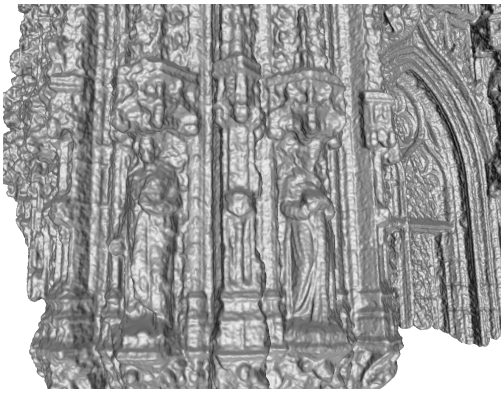
We have presented a fully geometric efficient *Lagrangian* solution for triangular mesh evolutions able to handle topology changes in an *intuitive and efficient* way. We have tested our method both in the context of mesh morphing in order to validate the method with challenging topological cases, as well as in the context of multi-view stereo 3-D reconstruction, where we have obtained top ranking results, comparable with state-of-the-art methods in the literature. Our contribution with respect to the existing mesh evolution methods is to provide a purely geometric mesh-based solution with proof of correctness, that does not constrain meshes and that allows for facets of all sizes as well as for topology changes.

[2] http://cvlab.epfl.ch/data/strechamvs/

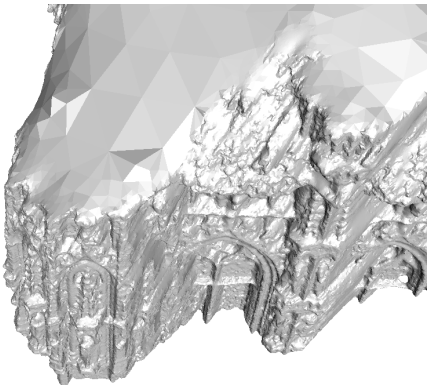
(a) Input Images (3000 × 2000)



(b) Front view

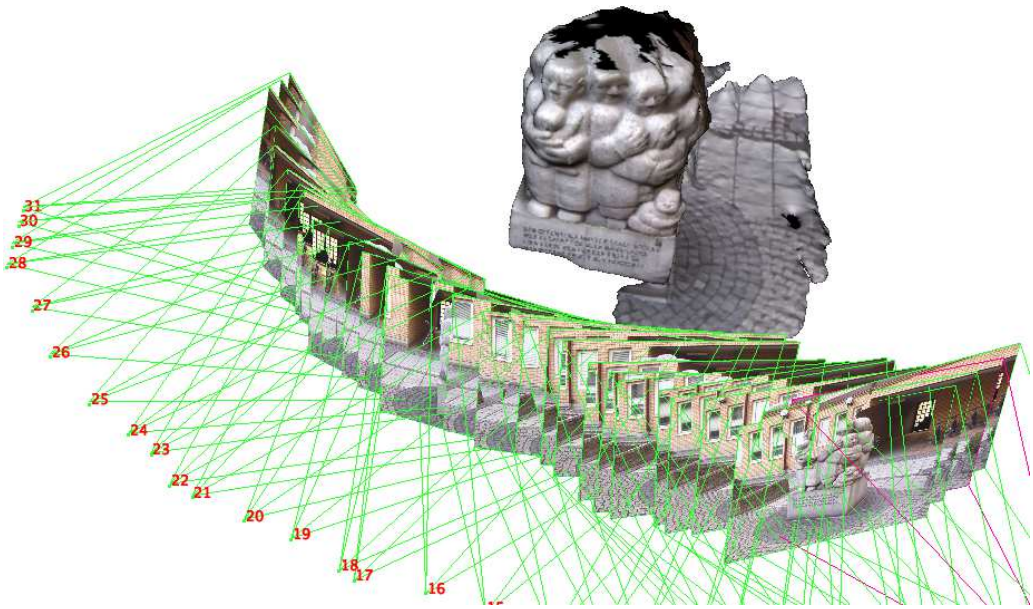


(c) Close-Up



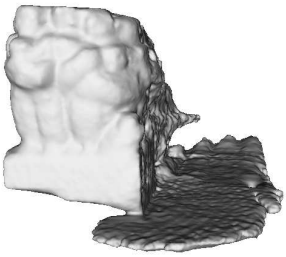
(d) Top view

Figure 5.14: Reconstruction results for the *Leuven* sequence. Invisible triangles are not further refined, as it can be observed in (d), maintaining the representation as compact as possible.

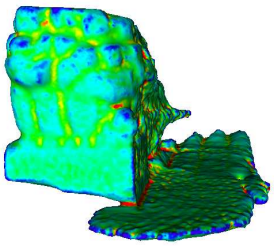


(a) Overview

(b) View 1 - Textured

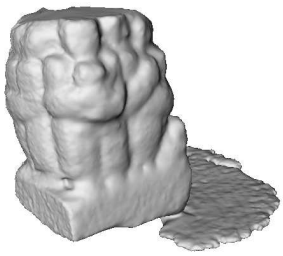


(c) View 1 - White

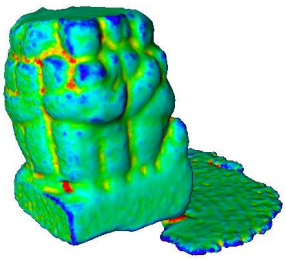


(d) View 1 - Curvature

(e) View 2 - Textured



(f) View 2 - White



(g) View 2 - Curvature

Figure 5.15: Reconstruction results for the Staty sequence.

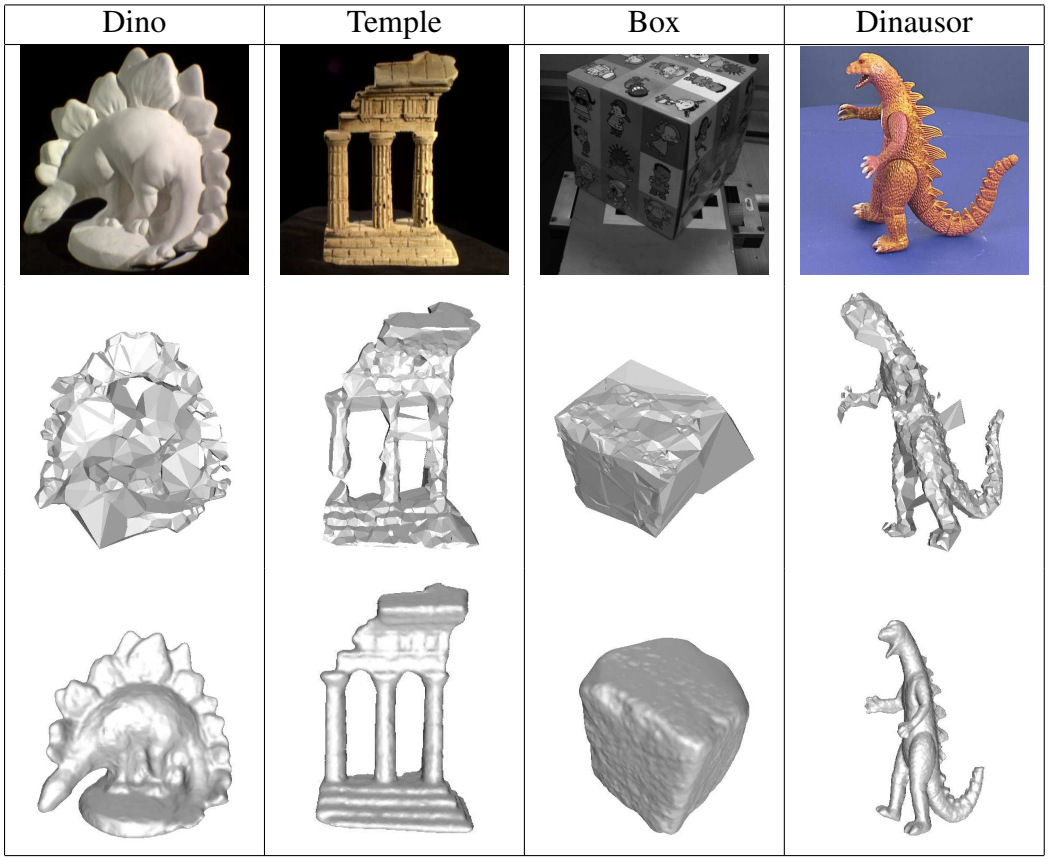


Figure 5.16: Additional dense reconstruction results demonstrating the connection with the previous chapter. Top Row: Sample Input Image; Middle: A rough mesh obtained from the 3-D reconstructed points using PowerCrust [12] Bottom: the final dense reconstruction after surface evolution using the method described in the current chapter.

Chapter 6

Camera-Clustering

In this chapter we propose a framework for piecewise mesh-based 3D reconstruction from a set of calibrated images. Most of the existing approaches consider all available images at once. However, this is not tractable with very large sets of cameras. Therefore, we use subsets of images and evolve parts of the surface corresponding to those images. Our main contribution is an approach to partition the camera images, either semi-automatic, through clustering, or user guided, via a geometric modeling interface. The sub-parts of the surface corresponding to camera subsets are independently evolved at multiple mesh resolutions. This allows to handle large scenes and to increase the mesh resolution in surface parts containing high levels of detail at reduced memory and computational costs. We demonstrate the versatility of our approach on different data sets and with different camera layouts. Finally, comparing the piecewise and global reconstructions with groundtruth, we find no significant loss in the overall reconstruction quality.

6.1 Introduction

Recent advances in multi-view 3D reconstruction from a set of calibrated cameras produced impressive results. The visual and measured quality is getting comparable to that of the laser scans. An issue of interest that naturally arises in this field is how to efficiently deal with scenarios where there are lot of images and, due to memory requirements, they cannot all be processed at the same time. In order to reduce the volume of image data we need to access simultaneously, we use subsets of the original image set and evolve the parts of the surface corresponding to those images by maximizing photo-consistency. The main contribution of our method is an approach to partitioning of the camera images which can be either semi-automatic, through clustering, or user guided, via a geometric modeling

interface. The sub-parts of the surface corresponding to the camera subsets are independently evolved at multiple mesh resolutions. This allows for an increase of the mesh resolution in surface parts containing high level of detail at reasonable memory and computational costs.

The problem of content-aware camera clustering and reconstruction by parts did not receive considerable attention in the past. Simon *et al.* [146] address an orthogonal problem to ours: scene summarization. In their scenario, they have a lot of images covering a scene and they are interested in the canonical views that can best describe it. They choose a representative exemplar from within each camera cluster, which is computed using visibility information for SIFT matches. There exist a a number of 3-D reconstruction methods [65, 54, 112] that can deal with large number of images, thus overcoming the apparent need for such a reconstruction by parts. As we shall see, they implicitly define heuristic camera clusters and they could benefit from the currently proposed algorithm. [112] casts the problem in a tracking framework and thus uses a temporal prior (sliding window). [65, 54] compute a set of sparse 3-D points from image correspondences, which are later on used to infer the full geometry. In order to reduce the search space for a given image/camera, the other image/camera is selected among the ones sharing the same viewing direction and rotation orientation. All these methods can benefit from our camera clustering method for special cases: revisiting the same sub-scene for [112]; camera panning scenarios for [65, 54].

A short review of the 3-D reconstruction methods was provided in Section 2.2.2. We will motivate the particular choice of the reconstruction algorithm, keeping in mind that the proposed camera clustering framework is very general and can thus be used in combination with any 3-D reconstruction method. *Variational methods* can adopt either an implicit surface (Eulerian) representation [123, 41, 131] or a mesh-based (Lagrangian) [109, 91, 80, 170, 130] point of view. They look for a surface which minimizes a global photo-consistency error function. The level-set implicit representation [131] requires dense regular sampling on a grid of the initial bounding volume, thus fixing the mesh resolution to the cell grid size. One advantage of such representations is the straightforward handling of topology changes at the cost of increased memory requirements. Evolving meshes directly calls for more elaborate schemes to handle topology changes and self-intersections, but offers a much more compact representation and can have an adaptive resolution compared to the implicit representations. Due to the smoothing energy terms, they tend to offer better resistance to outliers than dense multi-stereo approaches. The proposed recent advances in mesh-based methods, described in detail in Chapter 5, provide a solution to these problems and will be used. As opposed to the other Lagrangian methods, it does not constrain meshes to a fixed resolution and it allows for faces of all sizes. This approach to mesh

evolution, coupled with multi-resolution strategy on the surface parts, efficiently recovers objects of different complexity with the targeted precision on the more detailed surface parts.

We tested our approach on different data sets including single-compact objects, outdoor architectural sites filmed in high resolution, and a long synthetic sequence. Finally, we analysed quantitatively our results and compared our piecewise and global reconstructions to the laser scans, showing very little loss in the overall reconstruction quality.

In the reminder of the chapter we will describe our method, show the results of the experimental evaluation and finally conclude, talking about future work.

6.2 Method

Our objective is to evolve the complete surface by parts. This is an important aspect, if we want to reduce memory costs and computational time imposed when using all images at once. We rely on the recent mesh-based evolution method of Zaharescu et al. [170], which efficiently handles mesh topological changes and allows meshes with variable facet sizes. The mesh is evolved in parts over time by minimizing the photo-consistency error function proposed by Pons et al. [131]. For more details, consult [170], [131]. The mesh parts to be evolved are defined according to the partitioning algorithm discussed below. The camera clustering method that will be presented can work in combination with any 3-D reconstruction algorithm.

Camera Clustering. In general, if the positioning of the cameras is arbitrary and the rough initial geometry is known we can cluster original camera set C into a given number k of camera subsets $C_m, m = 1..k$. To do this, we first recover the geometry of the object/scene at a coarse resolution from down-sampled images using all cameras $c_i, i = 1..N_c$. For each camera c_i, we name S_i the set containing all the vertices from the scene set S which are visible.

We then define an intuitive distance function between two cameras c_p and c_q as the cardinal of the symmetric difference of S_p and S_q:

$$d_S(c_p, c_q) = |S_p \triangle S_q| \tag{6.1}$$

$$= |(S_p \cup S_q) \setminus (S_p \cap S_q)| \tag{6.2}$$

In practice we use OpenGL depth maps [141] to evaluate the visibility and accumulate the information into the $N_v \times N_c$ visibility matrix defined as:

$$\Delta = \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,N_c} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,N_c} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N_v,1} & \beta_{N_v,2} & \cdots & \beta_{N_v,N_c} \end{bmatrix}$$

where $\beta_{i,j}$ is 1 or 0 depending on whether the j^{th} vertex is visible in the camera c_i.

Instead of using a coarse mesh, one could also use potential SIFT matches in the image and accumulate them in the visibility matrix Δ, as it has been proposed by [146].

Camera-based clustering consists of performing k-means clustering [17] on the columns of Δ, where k represents the number of desired camera sub-sets. Each of these columns represent one camera, encapsulating visibility information for all the mesh vertices. Using these binary vectors, computing the distance function we defined in (6.2) is equivalent to computing the sum of squared differences :

$$d_S(c_p, c_q) = ||\Delta(:, p) - \Delta(:, q)||_E^2 \tag{6.3}$$

Note that each 3-D surface point has its contribution in the distance function, based on whether it is visible in both cameras. This simple formulation takes into account the geometry of the object and the layout of the cameras implicitly, by using the visibility information. We present some of the clustering results in Figure 6.1. Please note how the clustering correctly delineates the parts of the objects that share less visibility information (the two sides of the dinosaur, or the facets of the temple).

Geometry-based clustering Alternatively, one could address the dual problem and perform clustering on the rows of the matrix Δ, thus on the geometry of the scene. [1] Once the vertex clusters have been obtained, the set of the most discriminant cameras for each cluster has to be selected. This is done in practice by a voting method, imposing a minimum camera voting threshold of α times the average score among the camera with positive votes within each cluster. Using this dual formulation implies some tuning the α parameter, but has the great advantage of allowing potential camera overlaps, meaning that the same camera might be used by different vertex clusters. We present some of the clustering results in Figure 6.2. We have chosen $\alpha = 0.90$ in the dino case and $\alpha = 0.70$ in the temple case.

[1] The normalized point coordinates and the normal information can be added to the Δ matrix in order to take more geometric information into account.

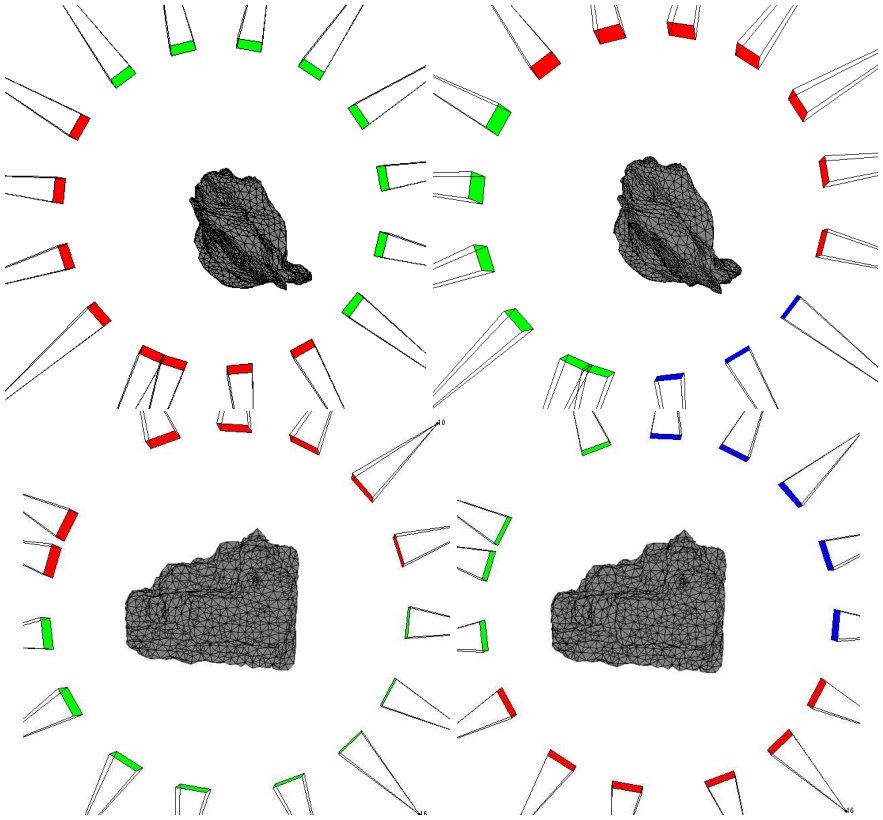


Figure 6.1: Illustration of *camera-based camera clustering* using two data sets, dino (first 2 images) and temple (next 2 images), with two and three clusters.

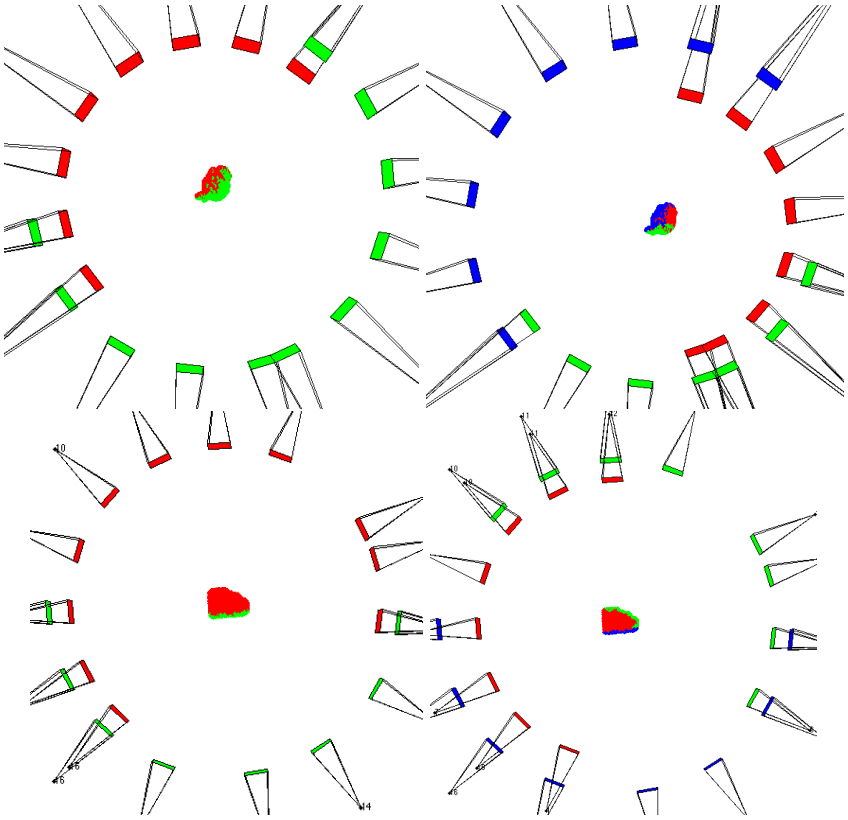


Figure 6.2: Illustration of *geometry-based camera clustering* using two data sets, dino (first 2 images) and temple (next 2 images), with two and three clusters.

Part-Based Surface Reconstruction. For each of the obtained clusters we run the algorithm described in [170], allowing only the vertices visible in the current camera cluster to evolve. In practice, we impose a minimum vertex visibility threshold γ. In order to avoid the issues related to merging partial reconstructions, we run one camera cluster at a time. The output of algorithm for one cluster is used as the input for the subsequent cluster. However, this approach comes at the expense of being unable to parallelize the approach in the current formulation. Alternatively, we could use algorithms such as [83] to merge the reconstructions and process all the clusters in parallel.

6.3 Results

We demonstrate the possibility of manually selecting vertices in a geometric modeling interface, in order to recover the surface regions of interest in high resolution. We also present the results of 3D reconstructions using our camera partitioning method. To demonstrate the versatility of our approach, we use different data sets, shown in Figure 6.3. When minimizing each subset, only the visible parts of the mesh are being sub-divided and minimized, while the others are blocked. We impose a minimum vertex visibility of $\gamma = 3$ in all cases.

(a) Dino (b) Parthenon (c) Temple

(d) Herz-Jesu (e) Fountain

Figure 6.3: Original images from the datasets used in our experiments.

6.3.1 User Guided Multi Resolution Scenario

The *User Guided Multi Resolution results* are presented on the "Herz Jesu" and the "Fountain" sequences of [151]. These experiments make use of user-defined regions of interest to increase the mesh resolution.

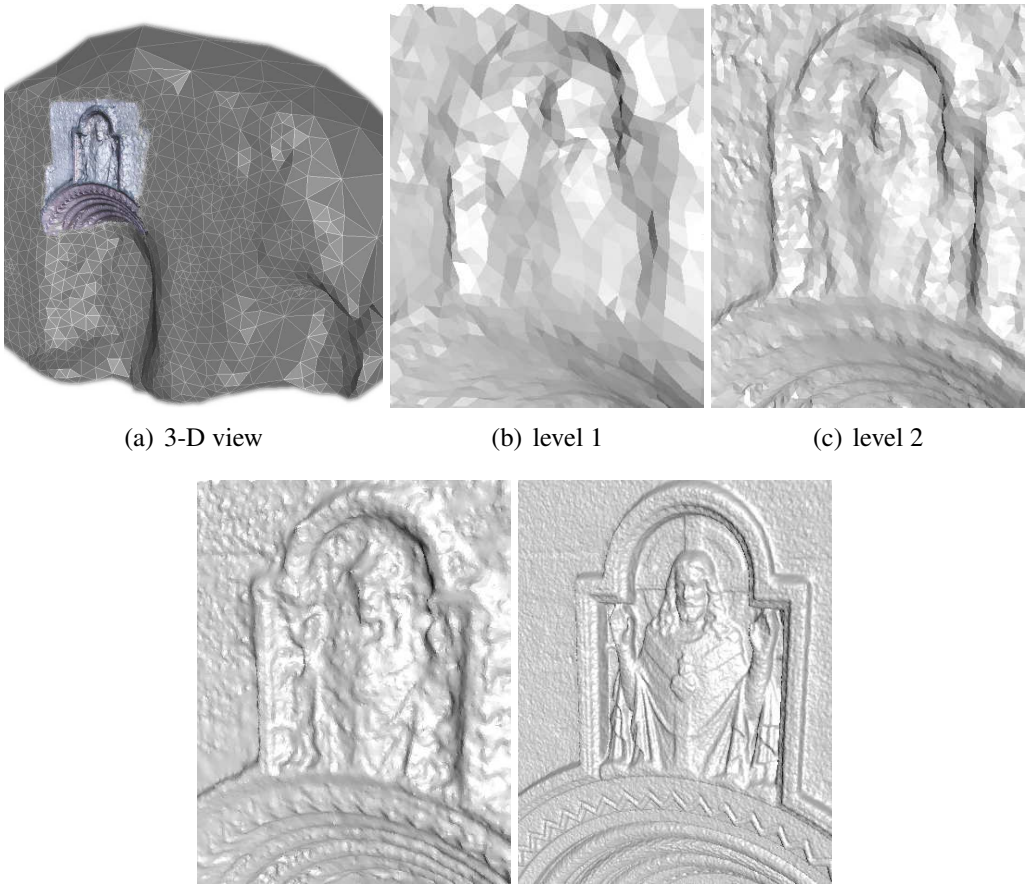
These sequences illustrate the interest of evolving directly a mesh representation of the reconstructed geometry when assisting a 3D artist in the task of visual modeling. The user can manually select a region of interest by selecting the corresponding vertices in the current approximation of the geometry and then ask the system for a further improvement of this part of the mesh. The higher resolution part can then be automatically evolved to maximize the photo-consistency accross the input image set. Virtual cameras are thus generated, representing the relevant input image sub-parts. In practice, cropping an image at coordinates (x_1, y_1, x_2, y_2) modifies the associated camera projection matrix by translating optical center by (x_1, y_1).

The Herz Jesu Sequence consists of 8 high resolution (3072 x 2048 pixels) pictures. The general view of the coarse reconstruction can be found in Figure 6.4(a). The scene was very interesting in the validation of our algorithm, because it involved different parts which had very different levels of detail. The wall can be represented by a coarser resolution mesh, whereas the door and the sculpted representation of Jesus above it are regions of interest that can benefit from a higher resolution reconstruction. The sculpture, in particular, is a region that a user might want to recover, but would not be able to quickly model it using simple geometric primitives. Our method allows to rapidly select the corresponding vertices and to let the algorithm maximize the photo-consistency. In addition, we have also obtained from the author groundtruth data for a part of the reconstruction, which was acquired via laser-scanning. The error measurements are presented in Table 6.1.

Level	Avg. DistanceError	Completeness (0.05m)	Avg.Edge Size	Avg.Edge Size	No. Triang.
Level 1	0.0270m	83.36%	0.1347m	21.04 pixels	2,136
Level 2	0.0177m	92.90%	0.0703m	10.98 pixels	8,592
Level 3	0.0164m	94.17%	0.0232m	3.62 pixels	82,490
Groundtruth	0.0000m	100.00%	0.0064m	1 pixel	1,693,914

Table 6.1: Information about Herz Jesu reconstructions. The errors are measured in meters. The completeness is measured with respect to a threshold of 0.05m.1 pixel corresponds to an edge size of 0.0064m.

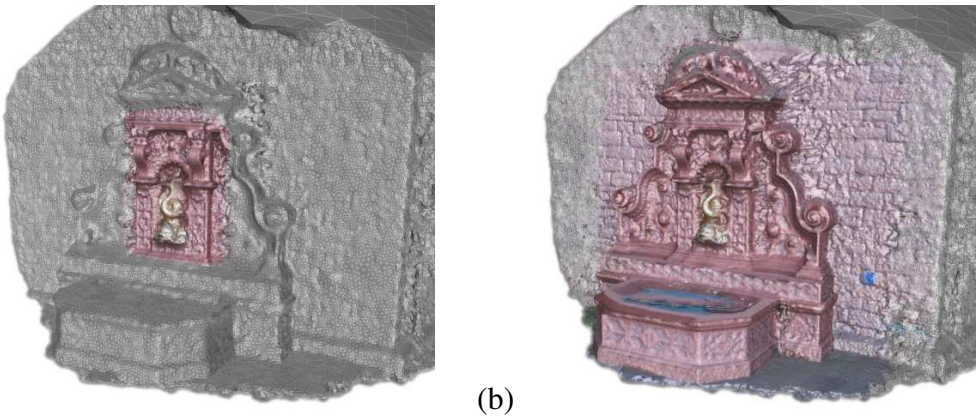
The Fountain Sequence consists of 11 pictures of size 3072 x 2048 pixels, taking up 64.4 Mb in compressed format. It involves very fine 3D details and was



(a) 3-D view (b) level 1 (c) level 2

(d) level 3 (e) groundtruth

Figure 6.4: The Herz-Jesu sequence



(a) (b)

Figure 6.5: The Fountain sequence. The parts shown in color are reconstructed in higher resolution. (a) Further minimization on the fish; (b) Further minimization on the whole fountain.

therefore a good stress test. We pushed the algorithm to a very high resolution of 4 pixels per triangle. We ran two tests on this dataset. The first test was to reconstruct the whole fountain at a high level of detail, leaving only the wall behind in a coarser state. The algorithm needed 894 minutes to finish. We then ran the algorithm on the fish sculpture only and got a result after 83 minutes. This validates our approach in the sense that evolving a subpart of the reconstructed geometry independently from the rest allowed us to stay away from swapping and other memory problems. In both cases, we started the algorithm from a coarser reconstruction that was performed with all images at half the original image size. The results are presented also in Table 6.2.

Level	Img. Input Size	Avg.Edge	Avg.Edge	No. Triang.	Time
Coarse	64.4Mb	0.0750 m	20.83 pixels	75,904	129 mins.
Close-up Fish	5.4Mb	0.0161 m	4.44 pixels	151,564	129 + 83 = 212 mins.
Close-up Fountain	40.6Mb	0.0160 m	4.44 pixels	660,540	129+894 = 1,023 mins.

Table 6.2: Information about Fountain reconstructions. 1 pixel corresponds to an edge size of 0.0036 m. The image input size value represents the total compressed size of the input images, which can be further sub-sampled, depending upon the resolution used.

6.3.2 Camera Partitioning

The camera partitioning algorithm is presented on two very different types of sequences. We first validate the method on a typical turntable situations, where

the object bounding volume projects inside all the images of the sequence. The Dino and Temple datasets are presented. We then present the Parthenon dataset, which involves one long image sequence covering a large object. In this case, each image only contains a small portion of the reconstructed geometry. We have used the camera-based clustering in all results shown below. The point-based camera clustering leads to very similar results. Due to the inherent overlap between views, we decided to use the simplest method.

Dino and Temple Sequence. These sequences were obtained from the Middleburry Multi-View Stereo dataset [142]. It consists of 47 images of size 640x480. The coarse surfaces were evolved from the visual hull using all down-sampled images at 320x240 resolution. The reconstruction results for two clusters are shown in Figure 6.6 and Table 6.3 (see [2] for more). Our proposed method does not lose significant accuracy with respect to the original method [170] (which uses all the cameras), while reducing the memory requirements in half and maintaining comparable time processing times.

Dataset / Paper	Temple Ring				Dino Ring			
	Acc.	Compl.	Mem.	Time	Acc.	Compl.	Mem.	Time
Zaharescu et al [170]	0.55mm	99.2%	1031MB	60min	0.42mm	98.6%	962MB	43min
Our method - cluster 1	0.62mm	98.5%	468MB	36 min	0.5mm	98.5%	483MB	33min
Our method - cluster 2			472MB	42 min			476MB	35min

Table 6.3: Middleburry 3-D Rec. Results. Accuracy: the distance d in mm that brings 90% of the result R within the ground-truth surface G. Completeness: the percentage of G that lies within 1.25mm of R. Memory: the amount or RAM used by the program. Time: the duration for the program to finish.

The parthenon sequence consists of 200 images of size 640x480. Each of these cameras covered only about 1/10th of the overall structure. This sequence is synthetic and was generated using Blender[3] and the textured models obtained from the Parthenon scuplture gallery website[4]. We have employed various camera cluster sizes, with $k = 2, 4, 8, 20$. The camera path can be observed in Figure 6.7 where we also show the camera clusters in different colors. The original surface was a parallelepiped of 4472 facets. In Figure 6.8 we show reconstruction results throughout the evolutions of different clusters.

We measured the reconstruction precision with respect to the groundtruth as shown in Table 6.4. As it can be observed, there is negligible loss in precision

[2] http://vision.middlebury.edu/mview/eval/

[3] http://www.blender.org/

[4] http://projects.ict.usc.edu/graphics/parthenongallery/index.html

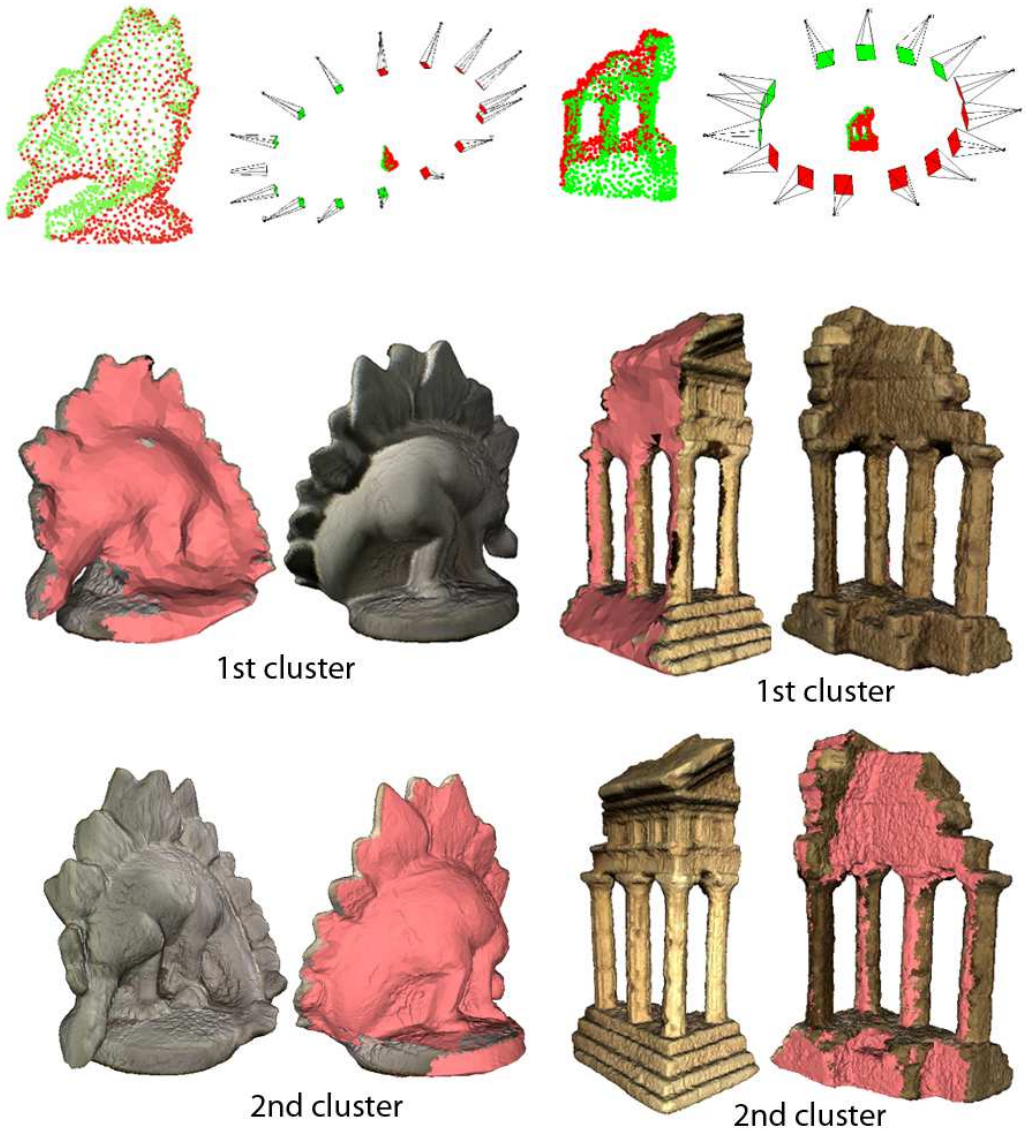


Figure 6.6: Dino and Temple Sequence reconstruction results. The top row presents the camera partitioning, whereas the middle and bottom row show the partial reconstructions for the two clusters. The invisible vertices within each cluster are coloured in light red. The reconstructions are made at 5 pixels per edge size.

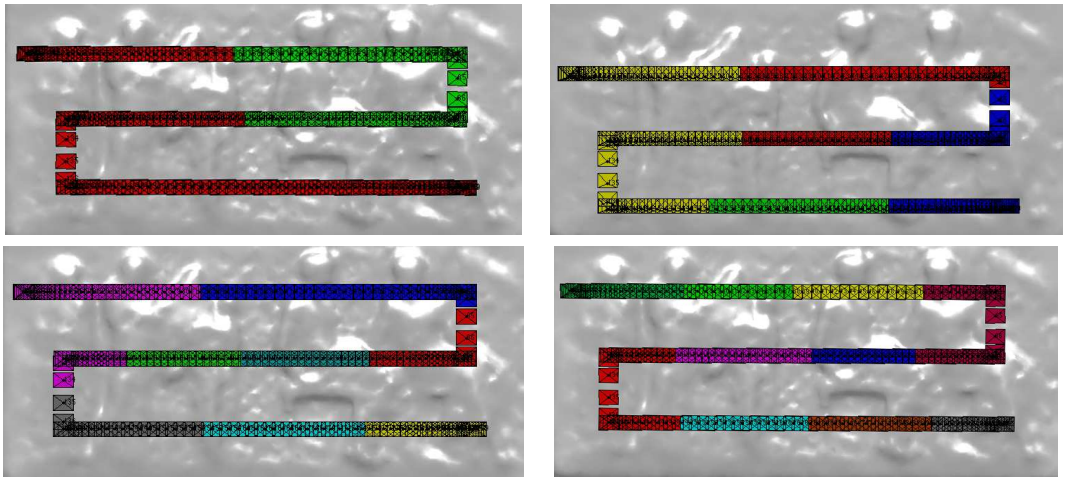


Figure 6.7: Different camera clustering for Parthenon sequence with the number of clusters being $k = 2, 4, 8, 10$. Cameras belonging to the same cluster are colored in the same color.

of 5mm when performing subset-based reconstruction versus when using all the cameras at the same time. One has to bear in mind that the laser error for the given distance is also around 5mm.

Level	Avg. Dist. Err.	Compl.. (0.05m)	Avg.Edge (m)	Avg.Edge (pixels)
21 Clusters - Low Res.	0.0344 m	76.98%	0.2528 m	16.20 pixels
All Cameras - Low Res.	0.0265 m	84.83%	0.2885 m	18.49 pixels
2 Clusters - High Res.	0.02006 m	92.50%	0.1629 m	10.44 pixels
4 Clusters - High Res.	0.0209 m	91.59%	0.1641 m	10.52 pixels
8 Clusters - High Res.	0.0201 m	92.35%	0.1599 m	10.25 pixels
21 Clusters - High Res.	0.0205 m	92.36%	0.1331 m	8.53 pixels
All Cameras - High Res.	0.0153 m	95.73%	0.1102 m	7.20 pixels
Groundtruth	0.0000m	100.00%	0.0841m	5.39 pixels

Table 6.4: The Parthenon reconstructions error measures, compared to the laser scan ground truth. The errors are measured in meters. The completeness (compl.) is measured with respect to a threshold of 0.05m. 1 pixel corresponds to an edge size of 0.0156m. Note there is negligible loss in precision when performing subset-based reconstruction versus when using all the cameras at the same time.

Virtual Fountain Sequence In the fountain sequence , since we are dealing with very high resolution images (3072 x 2048), we have generated virtual cameras such that the original image is cropped into a 2x2 grid (hence 4 virtual cameras for each real camera). We are pleased to report that, performing camera-

Figure 6.8: Parthenon reconstruction using 200 cameras and 21 clusters of cameras. We show partial reconstructions where parts of the surface are reconstructed using cameras belonging to one cluster at a time.

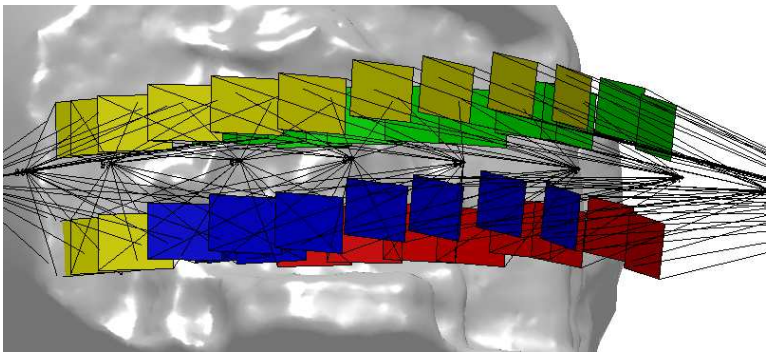


Figure 6.9: 4 Cluster View for the Virtual Fountain dataset.

subset clustering, the 4 correct subsets were found. Results can be observed in Figure 6.9. The reconstruction results per cluster are presented in Figure 6.10.

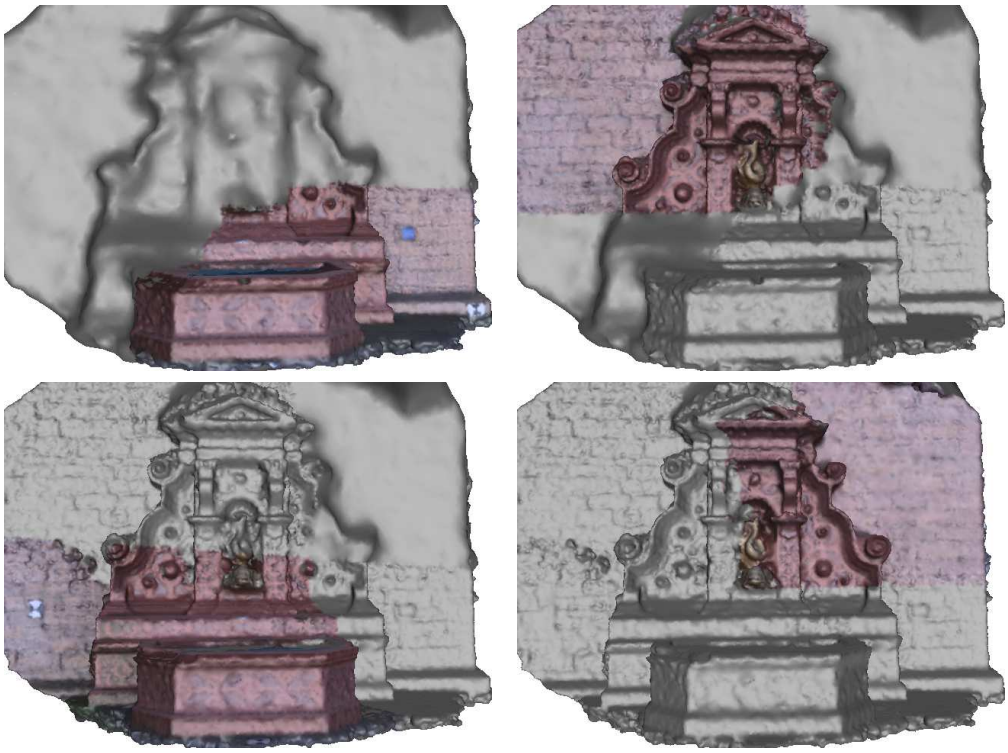


Figure 6.10: Results for the partial reconstructions in the virtual fountain scene.

Future Work. One other possible scenario that we plan on investigating is, instead of pre-generating virtual cameras and performing clustering, to generate the virtual cameras post-clustering, limiting the virtual cameras to the bounding boxes. Also, we plan on exploring automatic mesh segmentation methods that take into account more mesh properties, which will in turn allow for the selection/generation of the proper cameras. Finally, we plan on integrating a photo-consistency based threshold for adaptive mesh resolution. It would adaptively determine if a facets represents the geometry well enough, based on the reprojection error measure. It was not currently implemented due to time constraints and do to the fact that in practice we calculate only the derivative of the photo-consistency measure, not the measure itself.

6.4 Conclusion

In this chapter we addressed the problem of piecewise 3D surface reconstruction from multiple calibrated cameras. We showed that, starting from the coarse initial geometry, the original set of cameras can be partitioned into a number of camera subsets, each of which is observing a part of the surface to reconstruct. Independent reconstructions of surface parts require less memory than when using all cameras as in global approaches. We also showed the possibility of using these techniques in a graphical modeling interface, when regions of interest have to be reconstructed in high resolutions. We have demonstrated that the proposed method does not lose significant accuracy with respect to global methods, while offering several advantages with respect to the time and to the memory requirements.

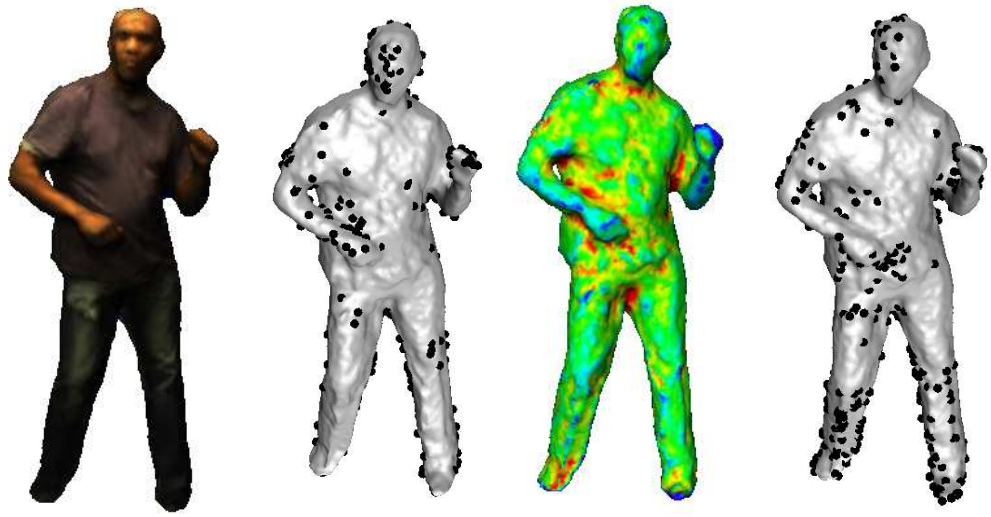
Chapter 7

3-D Mesh Descriptor

In this chapter we revisit local feature detectors/descriptors developed for 2-D images and extend them to the more general framework of scalar fields defined on 2-D manifolds. We provide methods and tools to detect and describe features on surfaces equiped with scalar functions, such as photometric information. This is motivated by the growing need for matching and tracking photometric surfaces over temporal sequences, as a number of recent approaches allow their estimations using multiple cameras. To this purpose we propose a 3-D feature detector (MeshDOG) and a 3-D feature descriptor (MeshHOG) for uniformly triangulated meshes. The descriptor is robust to changes in rotation, translation, and scale and it is able to capture the local geometric and/or photometric properties in a succinct fashion. Moreover, the descriptor is defined generically, it is valid with any scalar function, e.g., local curvature. Results with rigid and non-rigid mesh matching demonstrate the interest of the proposed framework. Lastly, it is shown how the approach integrates directly within a mesh tracking framework.

7.1 Introduction

The detection, characterization, and matching of various 2-D or 3-D features from visual observations is of great importance for a large variety of applications such as modeling, tracking, or recognition, indexing, etc. The vast majority of existing methods detect features using either photometric information available with 2-D images or geometric information available with 3-D surfaces. However, recent progress in image based 3-D modeling and rendering allows to recover both photometric and geometric information from multiple images [142]. Whenever such models are available, photometric 2-D features or geometric 3-D features, if



(a) (b) (c) (d)

Figure 7.1: The feature detection method described in this chapter can be applied to any scalar function defined over a 2-D manifold such as the meshed surface shown here: photometric data (a) and associated points of interest (b); mean surface curvature (c) and the detected features (d).

taken separately, have limited informative capabilities with respect to the potential richness of the data. Consequently, even though observations from several viewpoints are available, matching features is still challenging in several situations. This is the case, for example, with deformable and/or articulated objects, since image appearance is only partially robust to motions and geometric properties by themselves are not always robust, e.g., the topology of the model can change considerably with varying object poses. Therefore, we believe that photometric and geometric information need to be handled in a consistent and simultaneous manner. To this purpose, we observe that photometric 3-D models can be viewed as scalar functions defined over 2-D manifolds and, as such, represent a generalization of planar image domains to non-planar domains. We can thus build on the existing theories and body of work on extracting image feature, and investigate their extensions to 2-D manifolds.

The contribution of this chapter is twofold: first we develop a methodology for feature-based characterization using operators acting on scalar functions defined over 2-D manifolds; second, we derive a novel family of interest point detectors and descriptors that take into account both the surface geometry and the photometric information. To this aim, operators such as the *discrete convolution* and the *discrete gradient*, are defined for scalar functions on discrete surfaces, i.e.,

meshes, thus taking into account both the functions' values as well as the surfaces' geometry. Based on these operators, a new interest point detector and a new local descriptor are introduced, namely MeshDOG and MeshHOG. MeshDOG is a generalization of the DOG operator [102, 97] and it seeks the extrema of the Laplacian of a scale-space representation of any scalar function defined on a discrete manifold. MeshHOG is a generalization of the histogram of oriented gradients (HOG) descriptor recently introduced for describing 2-D images [33]. The new descriptor is defined with respect to the measurements available at each of the discrete surface's vertices and it can work with features such as colour, curvature, or geodesic integral, among others.

As is the case with the more classical image operators, detectors and descriptors are not uniquely defined over surfaces and MeshDOG and MeshHOG were chosen in light of their quasi-invariance to transformations such as rotation and scale. In addition, they exhibit a number of attractive properties:

- There are no perspective distortions, since computations are achieved in 3-D;
- There are no false detections due to occlusions;
- The descriptor captures both the local 3-D geometry and the local gradient information of the scalar function;
- Within a multiple-camera setting, the descriptor can fuse the photometric information coming from different images in order to provide more robust image-invariant photometric information.

The organization of the chapter is as follows. Section 7.2 discusses related works. Section 7.3 describes the mathematical formulation allowing us to build a number of operators on discrete manifolds. Section 7.4 and 7.5 introduce the local feature detector and descriptor, respectively. Section 7.6 presents and discusses the results, before concluding in section 7.7.

7.2 Related Work

Photometric functions over planar domains (local image features): Developing robust 2-D features, invariant under changes in illumination, viewpoint, scale and orientation has been one of the long term research goals in computer vision. Currently, SIFT [97] and HOG (histogram of oriented gradients) [33] are

among the most widely used descriptors for their robustness to the transformations just cited. Interest points may coincide to the extrema of the Laplacian of the photometric function, and they are detected at various resolution scales using the difference of Gaussians (DOG) approximation of the Laplacian, see [114] for a detailed review. Alternatively, spatio-temporal descriptors have also been proposed [166, 85], by considering the 3-D spatio-temporal volume defined by a short image sequence over time. Such space-time features can be seen as local features defined over 3-D grids. We extend the DOG operator to non-planar surfaces instead of dealing with volumetric grids.

Geometric functions over surfaces (local geometric features): 3-D spin images [78] and 3-D shape contexts [88, 51] are among the most successful surface descriptors. These are descriptors that rely solely on the surface geometry. See [154, 30] for a detailed survey. Typically these descriptors characterize the neighbourhood of a specified surface region. A number of methods have been proposed for automatic identification of interest regions on surfaces, taking into account geometrical features. Scale-space extrema based on the averaged mean curvature flow are proposed in [140]. Alternatively, [120] defines the scale space in a planar parametrization of the surface using the normal information and searches for the extrema. Photometric information is not taken into account by these methods.

Photometric functions over surfaces (local augmented surface features): In [167] a SIFT-based descriptor on 3-D oriented patches is proposed, i.e., VIP (Viewpoint Invariant Patches), which was used for 3-D model matching. It constitutes a first attempt to devise a descriptor that includes both geometry (normal orientation) and photometric information. In [148] the authors propose a concatenated surface descriptor taking into account both geometry (a region descriptor based on geodesic-intensity histograms), and photometric information (edge and corner descriptors that take into account the local isometric mapping to $\mathbb{R}^2$). The approach proposed in this chapter is similar in spirit to [167], but extended to consider full 3-D gradients and histograms.

Many applications make use of local features, in particular in the context of surfaces: surface registration, non-rigid shape matching and object recognition. For instance [121] proposes an image-based descriptor using the local $\mathbb{R}^2$ embedding of the curvature information on the mesh in order to perform surface registration. Also a recent number of works, e.g. [55, 6, 34, 160], address the non-rigid mesh matching problem using observations from multiple views. The vast majority of the proposed methods (the only notable exception being [55]) uses both geometric information extracted from surfaces and photometric data available with images. The latter is first extracted using 2-D image descriptors (such as SIFT [97]), and subsequently backprojected onto the mesh. This sparse description is generally used to bootstrap dense matching. Surface descriptors may well be used

for 3-D object recognition, as it has been already done in [143] using the Princeton shape benchmarking database [1]. Our work contributes to these efforts by improving the 3-D local feature extraction and description methodologies, as required by an increasing number of challenging applications.

7.3 Problem formulation

Let $\mathbb{S}$ denote the set of all possible discrete parametrizations of the admissible 2-D manifolds in $\mathbb{R}^3$. We will consider in particular *uniformly sampled triangulated meshes* $S \in \mathbb{S}$, namely meshes whose facets are triangles of approximately the same size and whose vertices' valence is close to 6. We notice that such an uniform mesh can be obtained from a non-uniform mesh through simple mesh operations, as proposed in [87]. This absolves us of the necessity of complex techniques that ensure proper samplings of scalar fields over S, while keeping generality. It is interesting to notice that an image can be viewed as a "flat" uniformly sampled mesh, i.e., a grid of vertices with valence 4 and whose facets are squares or rectangles.

S can also be viewed as a graph $S(V, E)$, where $V = \{v_i\}_{1 \leq i \leq N}$ is the set of mesh vertices and $E = \{e_{ij}\}$ is the set of mesh edges between adjacent vertices. We denote by e_{avg} the average edge length. We associate a 3-D point $\mathbf{v} \in \mathbb{R}^3$ with each vertex v. The ring of a vertex $rg(v, n)$ is the set of vertices that are at distance n from v on S, where the distance n is the minimum number of edges between two vertices. Thus $rg(v, 0)$ is v itself and $rg(v, 1)$ is the set of direct neighbours of v (see Figure 7.2). The neighbourhood $N_n(v)$ is then the set of rings $\{rg(v, i)\}_{0 \leq i \leq n}$. We further denote $\overrightarrow{\mathbf{n}}_v$ the normal direction at v computed as the average direction of the normals of the triangles incident to v.

We consider a scalar function $f : \mathbb{S} \rightarrow \mathbb{R}$. In order to be able to estimate discrete gradient information, we recall the definition of the directional derivative of a scalar function on a manifold:

Definition 1 (Directional Derivative) *Let $\nabla_S f$ denote the gradient operator of f on S, the directional derivative of f at $v \in S$ is defined as:*

$$D_{\overrightarrow{u}} f(\mathbf{v}) = \nabla_S f(\mathbf{v}) \cdot \overrightarrow{u}, \tag{7.1}$$

for any direction $\overrightarrow{u}$ in the tangent plane of S at v.

Using the fact that up to first order: $f(\mathbf{v}_j) - f(\mathbf{v}_i) = \nabla_S f(\mathbf{v}_i) \cdot (\mathbf{v}_j - \mathbf{v}_i)$ around v_i, we propose the following definition:

[1] http://shape.cs.princeton.edu/benchmark/

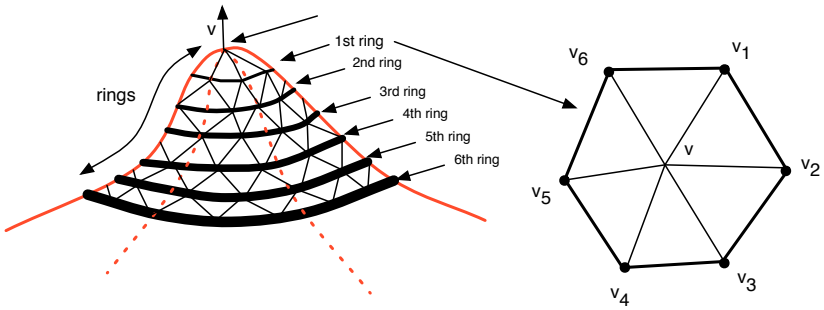


Figure 7.2: A vertex v and its rings (left) and the first ring of v (right).

Definition 2 (Discrete Directional Derivative) *The discrete directional derivative of f is defined as:*

$$D_{\overrightarrow{e_{ij}}} f(\mathbf{v}_i) = \frac{1}{||\overrightarrow{v_i v_j}||}(f(\mathbf{v}_j) - f(\mathbf{v}_i)), \tag{7.2}$$

$\forall e_{ij} \in E$ *and where* $||\overrightarrow{v_i v_j}|| = ||\mathbf{v}_j - \mathbf{v}_i||$.

$\nabla_S f(\mathbf{v}_i)$ is by definition a vector in the tangent plane at v_i and the above definition allows us to estimate its directional values around v_i. Hence, two such non-null local directional gradients are, in principle, sufficient to estimate the gradient $\nabla_S f(\mathbf{v}_i)$ at v_i. This is a generalization of the classical way of computing gradients in the image using two orthogonal directions. In practice however, we prefer to use all the directional gradients provided by the first ring of a vertex: indeed, this redundancy guarantees a more robust operator:

Definition 3 (Discrete Gradient) *the gradient operator $\nabla_S f(\mathbf{v}_i)$ of f at $v_i \in S$ is defined as:*

$$\nabla_S f(\mathbf{v}_i) = \sum_{v_j \in rg(v_i,1)} (w_{ij} D_{\overrightarrow{e_{ij}}} f(\mathbf{v}_i))\, \overrightarrow{u_{ij}}, \tag{7.3}$$

where w_{ij} weighs the contribution of $D_{\overrightarrow{e_{ij}}}$ and $\overrightarrow{u_{ij}}$ is the normalized projected direction of $\overrightarrow{v_i v_j}$ in the tangent plane at v_i.

The weights w_{ij} should be chosen in order to balance the contributions of the local directional derivatives with respect to their associated directions in the tangent plane. Assuming that S is uniformly sampled and thus that neighbours around v_i are equally spaced we get: $w_{ij} = \frac{1}{val(v_i)}$ where $val(v_i)$ is the valence of v_i. For

non uniformly sampled meshes, the weights are a function of the angles between the directions $\overrightarrow{u_{ij}}$ around v_i in the tangent plane at v_i[2].

Finally, we define the discrete convolution operator on a mesh:

Definition 4 (Discrete Convolution). *The convolution of the function f with a kernel k is:*

$$(f * k)(v_i) = \frac{1}{K} \sum_{v_j \in N_n(v_i)} k(||\overrightarrow{v_i v_j}||) f(\mathbf{v}_j), \tag{7.4}$$

where the kernel weighs the participation of neighbouring vertices v_j as a function of their distances from vertex v_i and $K = \sum_{v_j \in N_n(v_i)} k(||\overrightarrow{v_i v_j}||)$ is a normalization factor. Notice that, as for the discrete gradient, we assume a uniformly sampled mesh and thus that contributions of neighbouring vertices v_j in the above expression are equally weighted with respect to their spatial arrangements. Another remark is that, generally, we use the above definition with the first ring only, i.e., $n = 1$.

7.4 Feature Detector (MeshDOG)

Feature detection is comprised of three steps, as illustrated in Figure 7.3. First, the extrema of the function's Laplacian (DOG) are found across scales using a one-ring neighbourhood. Second, the extrema thus detected are thresholded. Third, the unstable extrema are eliminated, thus retaining those mesh locations exhibiting some degree of cornerness.

Scale Space Extrema. We propose a scale-space representation of scalar function f defined on a mesh. We consider the convolution operation on meshes (see Definition 4) using a Gaussian kernel, defined as:

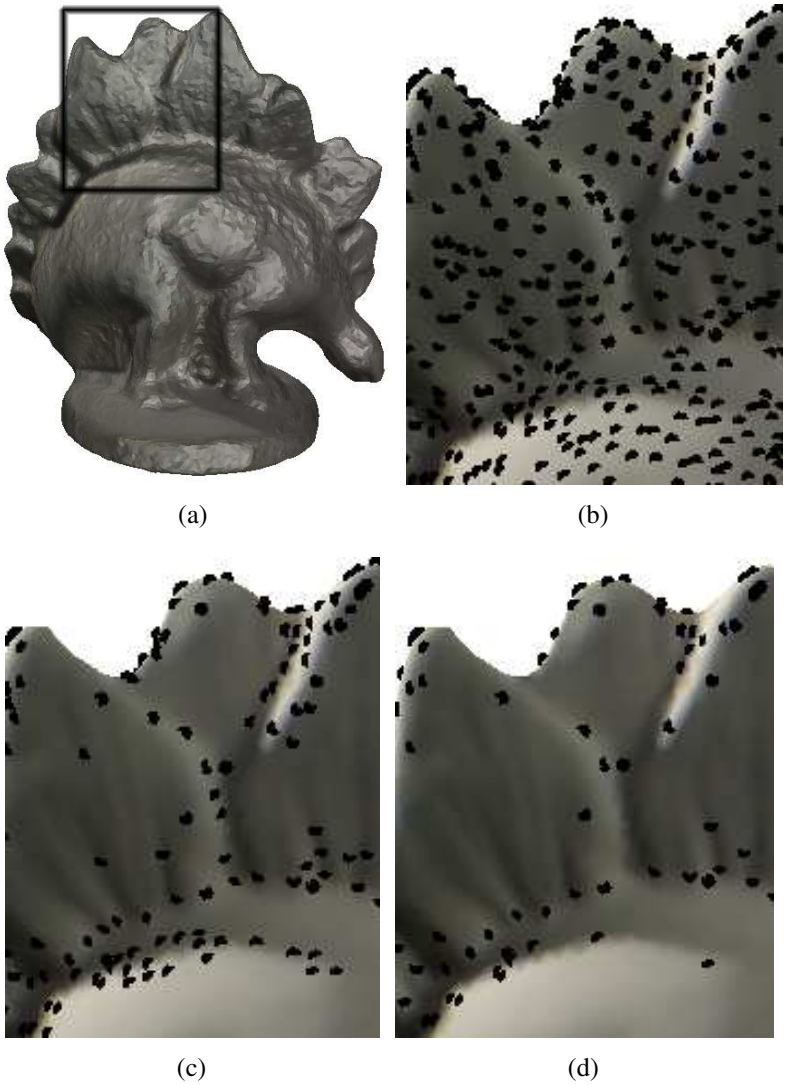
$$g_\sigma(x) = \frac{\exp\left(-x^2/2\sigma^2\right)}{\sigma\sqrt{2\pi}}.$$

The scale space of f is built progressively: $f_0 = f$, $f_1 = f_0 * g_\sigma$, $f_2 = f_1 * g_\sigma$, etc. Convoluted functions are subtracted, e.g., $DOG_1 = f_1 - f_0$, $DOG_2 = f_2 - f_1$, etc., in order to obtain the difference of Gaussian operator.

Such an example can be observed in Figure 7.4, where the model used is the *Stanford bunny* [3] and the mean curvature is chosen as a feature. Additional

[2] details are provided in an associated research report.

[3] http://www-graphics.stanford.edu/data/3Dscanrep/

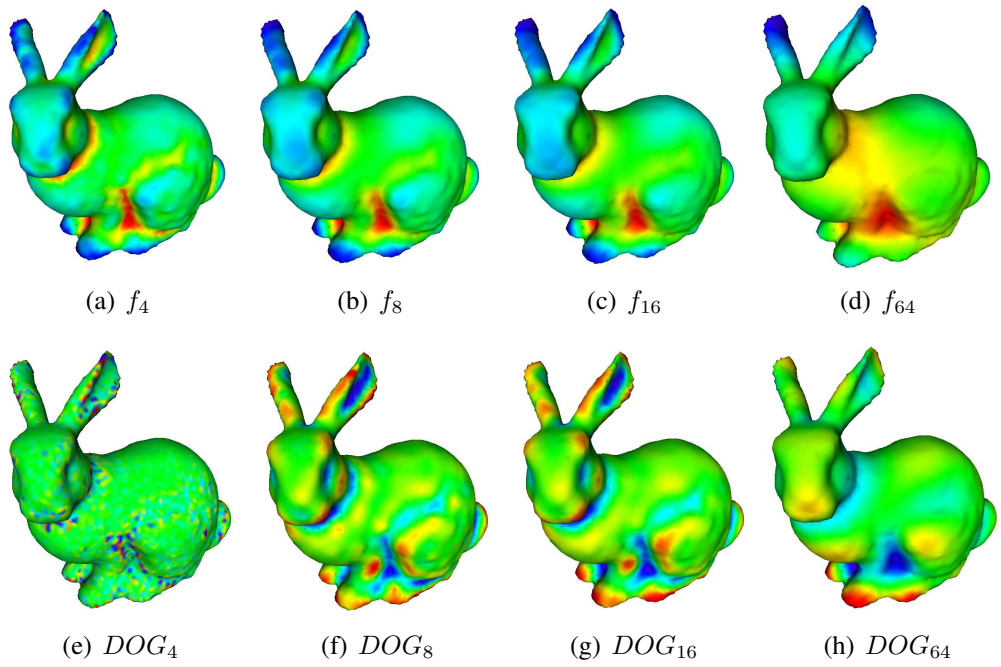


(a) (b)

(c) (d)

Figure 7.3: Feature detection shown with photometric data. (a) The original mesh has 27240 vertices. (b) There are 5760 extrema detected. (c) After thresholding there are 1360 vertices left. (d) 650 vertices with "cornerness" are eventually retained.

results are presented in Figures 7.5, 7.6 and 7.7 using the *pop2lock* sequence made available to us via the Surface Motion Capture project at the University of Surrey [149], where colour, mean curvature and guassian curvature have been used as features. An important observation is that, *when building the scale space, the mesh geometry does not change*, but the different scalar functions defined on the mesh, i.e. f_1, f_2, DOG_1, DOG_2.



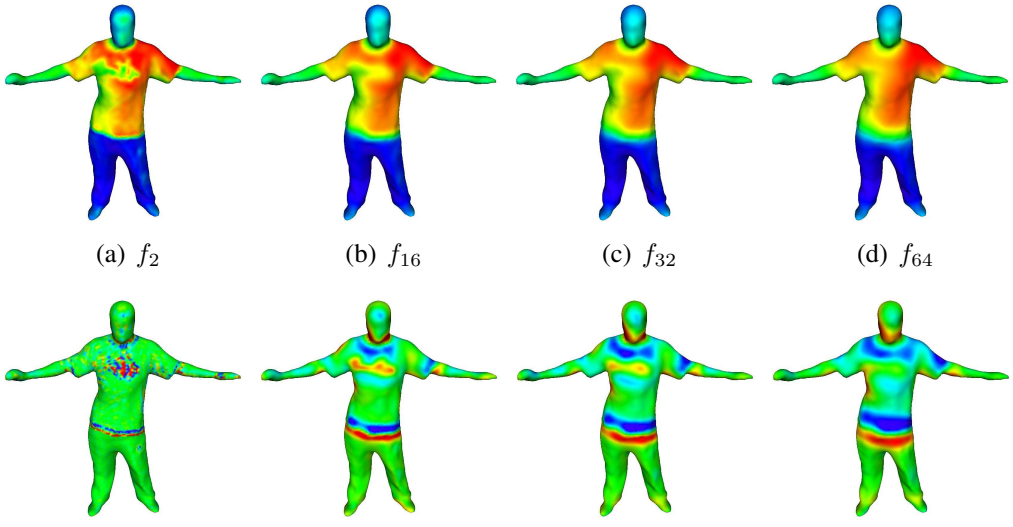
(a) f_4 (b) f_8 (c) f_{16} (d) f_{64}

(e) DOG_4 (f) DOG_8 (g) DOG_{16} (h) DOG_{64}

Figure 7.4: Scale space representation, where the shape is the *Stanford bunny* and the feature f is the mean curvature.

Each octave of $\sigma_0 = e_{avg}$ is subdivided in 3 steps, choosing $\sigma = 2^{\frac{1}{3}}\sigma_0 = 2^{\frac{1}{3}}e_{avg}$. In the 2-D image case, after covering an octave, the original image is sub-sampled first, before continuing to the next octave. In order to obtain the similar effect on triangular meshes, mesh simplification operations [93] can be performed, by imposing the new average edge to be $e_{avg}*1.5$. In order to cover the new octave, the σ_0 should be chosen to reflect the new average edge length. Due to the cost of the mesh simplification operation, alternatively, we chose not to simplify the mesh, but to continue convolving with the same original kernel, keeping in mind that, in order to cover another octave, twice as many convolutions are necessary.

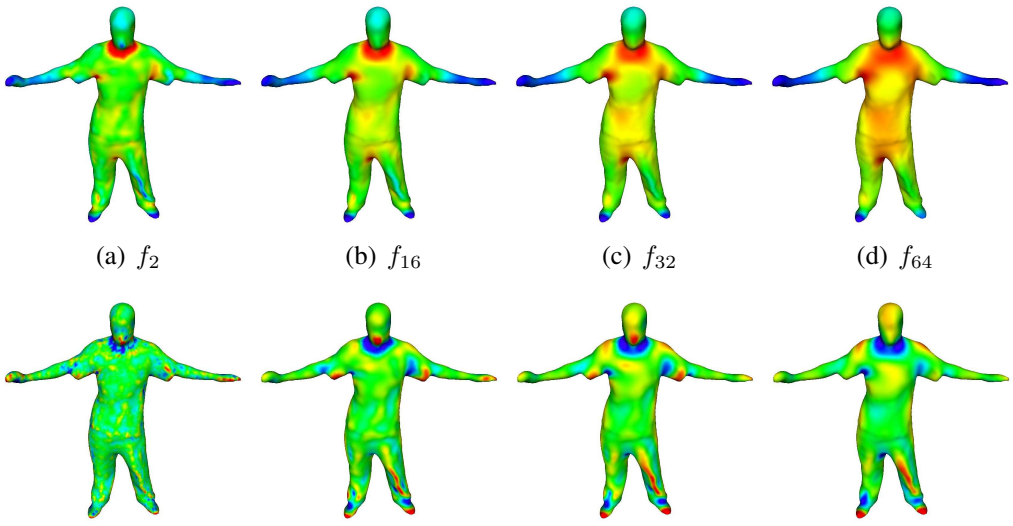
The feature points are selected as the maxima of the scale space across scales, followed by non-maximum-suppression, using the one ring neighbourhood, in the current and in the adjacent scales.



(a) f_2 (b) f_{16} (c) f_{32} (d) f_{64}

(e) DOG_2 (f) DOG_{16} (g) DOG_{32} (h) DOG_{64}

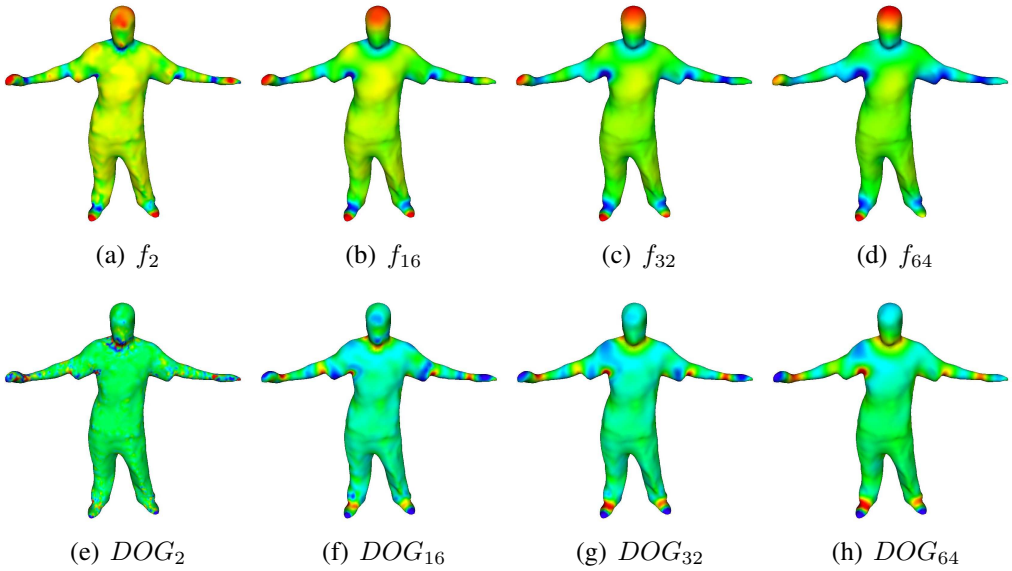
Figure 7.5: Scale space representation, where the shape is the *pop2lock* frame 30 and the feature f is the *colour*.



(a) f_2 (b) f_{16} (c) f_{32} (d) f_{64}

(e) DOG_2 (f) DOG_{16} (g) DOG_{32} (h) DOG_{64}

Figure 7.6: Scale space representation, where the shape is the *pop2lock* frame 30 and the feature f is the *mean curvature*.



(a) f_2 (b) f_{16} (c) f_{32} (d) f_{64}

(e) DOG_2 (f) DOG_{16} (g) DOG_{32} (h) DOG_{64}

Figure 7.7: Scale space representation, where the shape is the *pop2lock* frame 30 and the feature f is the *gaussian curvature*.

Thresholding. From the extrema of the scale space, only the top $\beta = 5\%$ of the maximum number of vertices are being considered, sorted by magnitude. We have chosen a percentage value versus a hard value threshold in order to keep the detector flexible, no matter which feature is being considered, without the need for normalization.

Corner Detection. Additionally, in order to eliminate more non-stable responses, we retain the features that exhibit corner characteristics. As proposed in [97] this can be done using the Hessian operator: :

$$\mathbf{H}(v) = \begin{bmatrix} d_{xx}(v) & d_{xy}(v) \\ d_{yx}(v) & d_{yy}(v) \end{bmatrix}, \tag{7.5}$$

where d_{xx}, d_{xy} and d_{yy} are second partial derivatives. We estimate them by applying the definition of directional derivatives (7.1) twice, e.g. $d_{xy} = \nabla_S D_{\overrightarrow{x}} f(\mathbf{v}) \cdot \overrightarrow{y}$, where the gradient is computed using (7.3). The directions $\overrightarrow{x}$ and $\overrightarrow{y}$ represent here a local coordinate system in the tangent plane of v, typically the gradient direction for $\overrightarrow{x}$ and its orthogonal direction for $\overrightarrow{y}$. The ratio between the largest λ_{max} and the lowest λ_{min} eigenvalues of the Hessian matrix is a good indication of a corner response. We typically use $\lambda_{max}/\lambda_{min} = 10$ as a minimum value to threshold responses.

7.5 Feature Descriptor (MeshHOG)

The descriptor for vertex v is computed using a support region, defined using a neighbourhood ring size r, as depicted in Figure 7.2. For each vertex from the neighbourhood $v_i \in N_r(v)$, the gradient information $\nabla_S f(v_i)$ is computed using (7.3). As a first step, a local coordinate system is chosen, in order to make the descriptor invariant to rotation. Then, a histogram of gradient is computed, both spatially, at a coarse level, in order to maintain a certain high-level spatial ordering, and using orientations, at a finer level. Since the gradient vectors are 3 dimensional, the histograms are computed in 3-D.

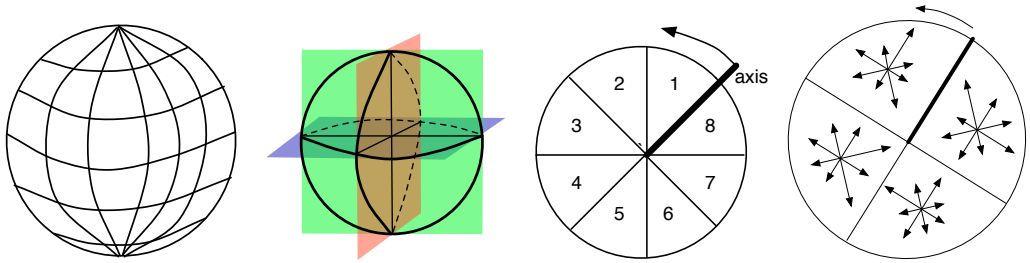
Neighborhood Size. The number of rings r for the support region is chosen adaptively based on a more global measure, such that the descriptor is robust to different spatial samplings and to scaling. The value of r is chosen such that it covers a proportion α_r from the the total mesh surface, where $\alpha_r \in (0, 1)$. By denoting A_S as the total surface area of the mesh S, which can be computed as the sum of all triangle areas, the ring size r is:

$$r = round\left(\frac{1}{e_{avg}}\sqrt{\frac{\alpha_r A_S}{\Pi}}\right), \tag{7.6}$$

assuming that the surface covering the ring neighbourhood can be approximated with a circle and that the mesh S is equally sampled, with the average edge size e_{avg}. In practice, we use an r corresponding to $\alpha_r = 1\%$.

Local Coordinate System. A local coordinate system can be devised using the normal $\overrightarrow{\mathbf{n}}_v$ and two other unit vectors, residing in tangent plane $\mathcal{P}_v$ of v. Given a unit vector $\overrightarrow{\mathbf{a}}_v \in \mathcal{P}_v$, the local coordinate system is given by $\{\overrightarrow{\mathbf{a}}_v, \overrightarrow{\mathbf{n}}_v, \overrightarrow{\mathbf{a}}_v \times \overrightarrow{\mathbf{n}}_v\}$. Vector $\overrightarrow{\mathbf{a}}_v$ is computed as the direction associated to the dominant bin in a polar histogram, with $b_a = 36$ bins. The histogram is computed by considering the projected vertices v_i in $\mathcal{P}_v$ and taking into account their gradient magnitudes. We weigh $||\nabla_S f(v_i)||$ by a Gaussian with $\sigma = e_{avg} r/2$, based on the geodesic distance from v. In order to reduce aliasing and boundary effects of binning, votes are interpolated bilinearly between neighbouring bins when computing the histograms. We use the same weighting and interpolation technique for any further binning.

Histograms. Instead of computing full 3-D orientation histograms, as proposed in [85], we project the gradient vectors to the 3 orthonormal planes, describing the local coordinate system. This provides us with a more compact representation of the descriptor. For each of the three planes, we compute a 2 level histogram. Firstly, the plane is divided in $b_s = 4$ polar slices, starting with an origin and continuing in the direction dictated by the right hand rule with respect



(a) (b) (c) (d)

Figure 7.8: a) 3-D Histogram - polar mapping used for creating histograms via binning of 3-D vectors; b) Choosing 3 orthogonal planes onto which to project the 3-D Histogram. c) Polar Coordinate system used for creating histograms via binning of 2-D vectors, shown in this example with 8 polar slices. d) Example of a typical spatial and orientation histograms, using 4 spatial polar slices and 8 orientation slices.

to the other orthonormal axis vector. When projected onto the plane, each vertex v_i will fall within one of the spatial slices. For each spatial slice, we compute orientation histograms with $b_o = 8$ bins for each of the projected gradient vectors $\nabla_S f(v_i)$ of the vertices v_i that projected onto that spatial slice, as shown in Figure 7.8(d).

Descriptor. The final descriptor is obtained by concatenating $b_s \times b_o$ histogram values for each of the three planes, followed by L^2 normalization.

7.6 Mesh Matching

We are validating the proposed detector and descriptor using a mesh matching approach. Let us consider two meshes S_1 and S_2 of the same object. The two meshes do not necessarily have the same number of vertices. Using the proposed approach, n_1 interest points are detected on S_1, which are characterised by descriptors $\mathbf{t}_{(1,i)}$, with $i \in [1..n_1]$. Similarly, n_2 interest points are detected on S_2, characterised by descriptors $\mathbf{t}_{(2,j)}$, with $j \in [1..n_2]$.

Matching. We use the following Greedy heuristic in order to select the a set of best matches. Let the Euclidean distance $d_{ij} = ||\mathbf{t}_{(1,i)} - \mathbf{t}_{(2,j)}||$ represent the matching score between descriptor $\mathbf{t}_{(1,i)}$ from surface S_1 and descriptor $\mathbf{t}_{(2,j)}$ from S_2. Now, for each descriptor $\mathbf{t}_{(1,i)}$, consider the best two matches $t_{(2,b(1,\mathbf{t}_{(1,i)}))}$ and $t_{(2,b(2,\mathbf{t}_{(1,i)}))}$ from S_2, in terms of the defined matching score, where the function $b(x, \mathbf{t}_{(1,i)})$ returns the index of the x^{th} best match among the descriptors of S_2. The

putative match $(\mathbf{t}_{(1,i)}, t_{(2,b(1,\mathbf{t}_{(1,i)}))})$ will be considered a good match only if it is significantly better than the second best match, that is if $d_{i,b(1,\mathbf{t}_{(1,i)})}\gamma > d_{i,b(2,\mathbf{t}_{(1,i)})}$, with $\gamma = 0.7$. Additionally, cross validation is performed, by checking that $t_{(2,b(1,\mathbf{t}_{(1,i)}))}$'s best match among the descriptors of S_1 is indeed $\mathbf{t}_{(1,i)}$. This is not meant to fully solve the matching problem, as would a global approach [148]. It is merely intended for validation and for evaluation of our detector and descriptor.

Datasets. In our evaluation we consider the following scenarios: (i) the two meshes are representations of the same rigid object, which can thus be aligned using a rotation, translation and scale; (ii) the two shapes are representations of the same non-rigid object, i.e. a moving person. In this context, we are introducing the datasets.

- Rigid Matching: we are considering reconstructions of the same object using different camera sets. In particular, we are using meshes obtained employing the method described in [170], using the publicly available datasets from the Middleburry Multi-View Stereo site [142]. The *Dino* datasets contains two meshes, one with 27240 vertices obtained from 16 cameras and the other of 31268 vertices generated from 47 cameras. Similarly, the *Temple* datasets contains two meshes, one with 78019 vertices obtained from 16 cameras and the other of 80981 vertices generated from 47 cameras.

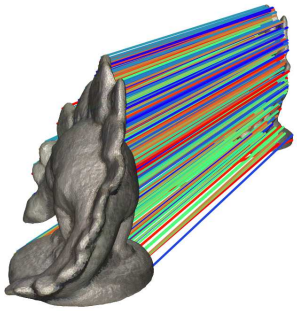
- Non Rigid Matching from Synthetic Data: we consider a synthetically generated dataset entitled *Synth-Dance* of a human mesh with 7061 vertices moving across 200 frames.

- Non-Rigid Matching from Real Data: additionally, we use frames 515-550 from the INRIA *Dance-1* sequence, where the same reconstruction method [170] was employed to recover models using 32 cameras. The models have vertices ranging between 16212 and 18332.

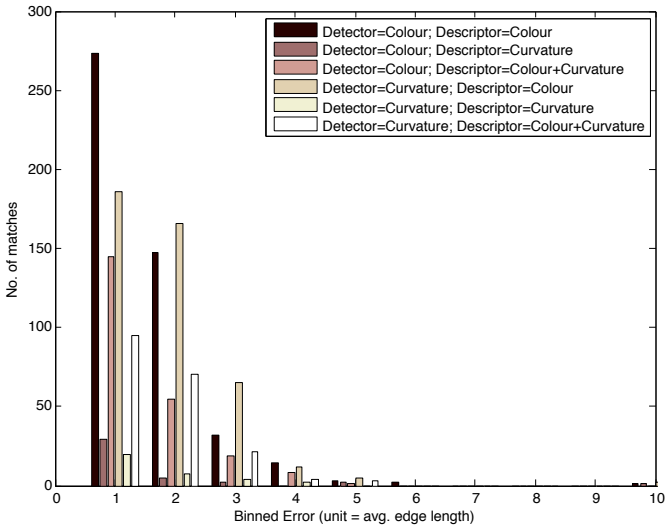
Photometric information. The colour of each vertex of the surface is computed by considering the median colour in the visible images. We assume that the colours of a vertex follow a non-Gaussian distribution, due to errors that can occur around occluding contours. In the *Synth-Dance* dataset the vertices are randomly coloured.

7.6.1 Rigid Matching

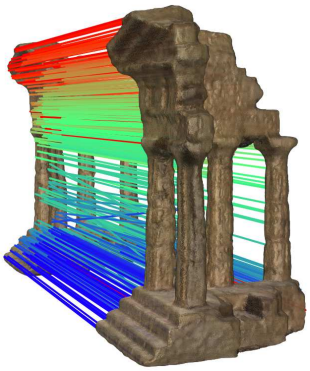
We present our results on the *Dino* and *Temple* datasets in Figure 7.9, where we have run tests where the colour and the mean curvature were used as features,



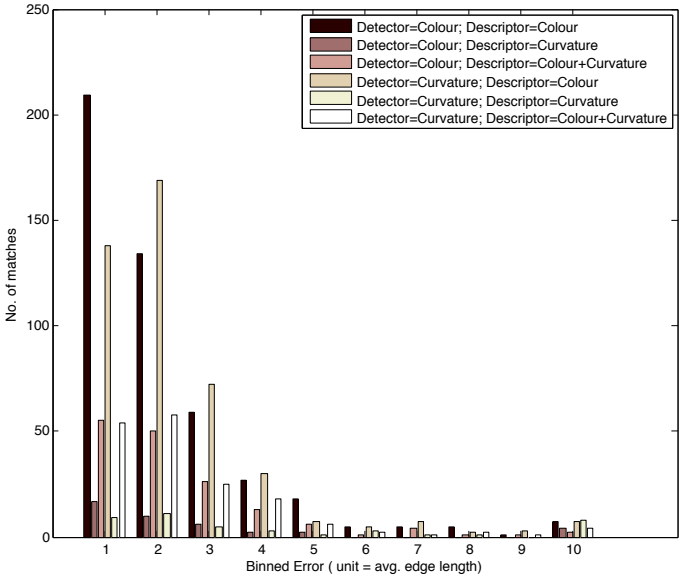
(a) Dino - Matches



(b) Dino - Errors



(c) Temple - Matches



(d) Temple - Errors

Figure 7.9: Rigid matching results - *Dino* and *Temple* datasets. (a) (c) Matching results when using the colour both as a detector and as a feature; (b) (d) Error distribution when using different combinations of features for both detection and matching.

as well as cases in which we have created a new descriptor by concatenating the MeshHOG descriptors for colour and mean curvature. In order to provide the reader with an intuition behind the different choices for features, we include Figure 7.10, which displays the colour, the mean curvature and the Gaussian curvature, as well as the above measure derivative magnitudes.

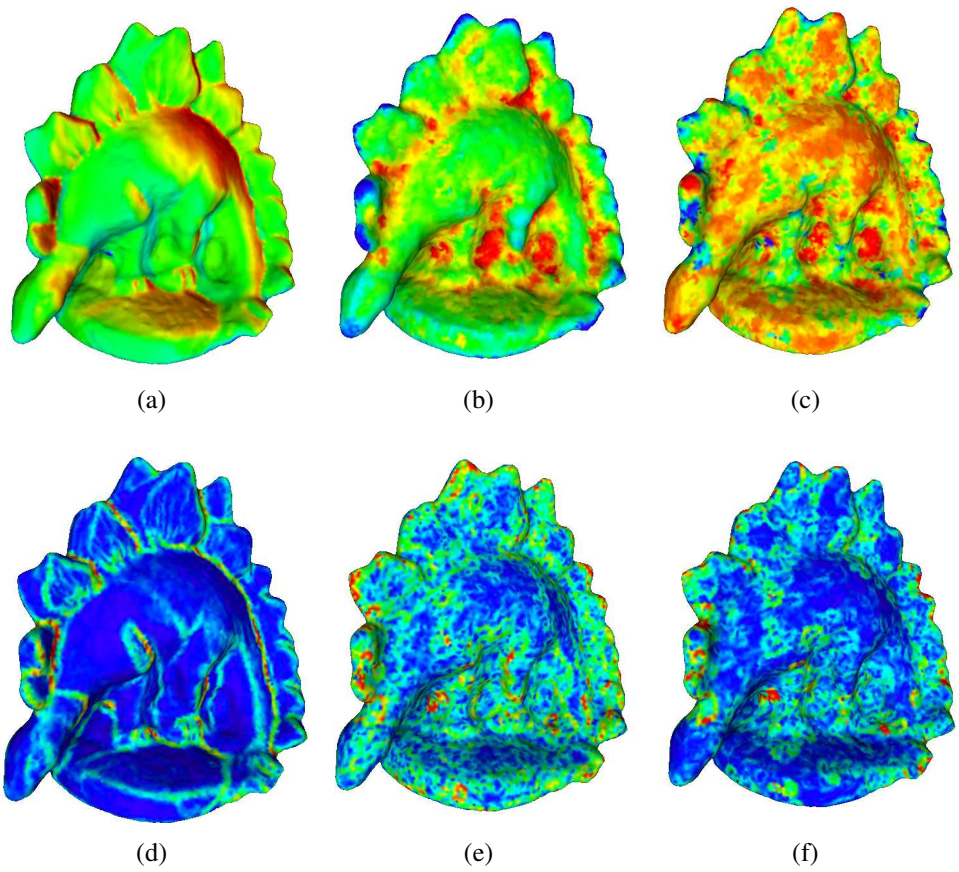


Figure 7.10: Different measures for the *Dino* dataset: (a) colour; (b) mean curvature; (c) Gaussian curvature; (d) colour gradient magnitude; (e) mean curvature gradient magnitude; (f) Gaussian curvature gradient magnitude.

The results are interesting. Even when just curvature is used for the descriptor, there seems to be enough discriminability to account for a number of correct matches varying between 10-30, depending on the detector and the dataset. Both the *Dino* and the *Temple* datasets are rather challenging, due to the fact that, at a first glance, they do not have a large number of distinguishing non-repetitive features in terms of their visual aspect. Additionally, it seems that using just the colour as a feature provides the best results in terms of the number of matches.

This is so, we can argue, because the descriptor inherently incorporates certain mesh geometry information by design of the operators.

These are the only results presented in the chapter where different features were used for the descriptor. All the other results are generated using colour information.

7.6.2 Non Rigid Matching

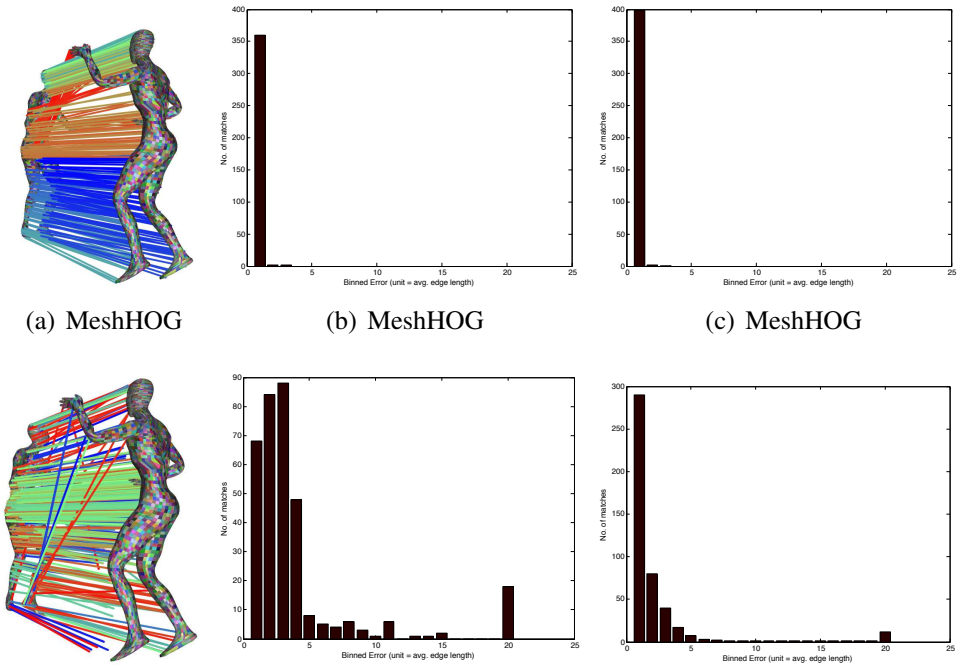
Comparison with Back-Projected 2-D Features. We present a comparison between the proposed mesh matching framework using MeshHOG descriptor with another framework, currently employed in a number of mesh matching methods (see Section 7.2), that uses back-projected image descriptors. In the image based framework, the matching is performed in the images and only then is back-projected onto the surface. In our comparisons, we used the SIFT image descriptor. When matching the two surfaces, only matches from the same cameras are considered. In order to be able to carry such a comparison for the *Synth-Dance* dataset, we have generated images for 16 virtual cameras, distributed in a circular pattern around the object.

Synthetic comparative results are presented in Figure 7.11. The mesh in the first frame was matched with the mesh at any of the other 199 frames across the sequence. As it can be observed, the MeshHOG descriptor generates very few false positives in comparison with the SIFT equivalent, clearly demonstrating the advantages of the proposed approach.

In addition, we present empirical results in Figure 7.12 for for the INRIA *Dance-1* sequence. As it can be observed, the second second best match ratio threshold $\gamma = 0.7$ tends to be more aggressive for SIFT. There are only 54 matched found using the SIFT back-projected method between frame 525 and 526, whereas MeshHOG finds 119 matches. Even when matching across distant frames (530 and 550), our proposed method finds 13 correct matches, versus the SIFT descriptor, that fails to find any match. It is to be expected, since most of the inter-frame matches are due to local creases formed by the clothes.The head is the only unique feature that can be robustly matched across time.

7.6.3 Resilience to Noise

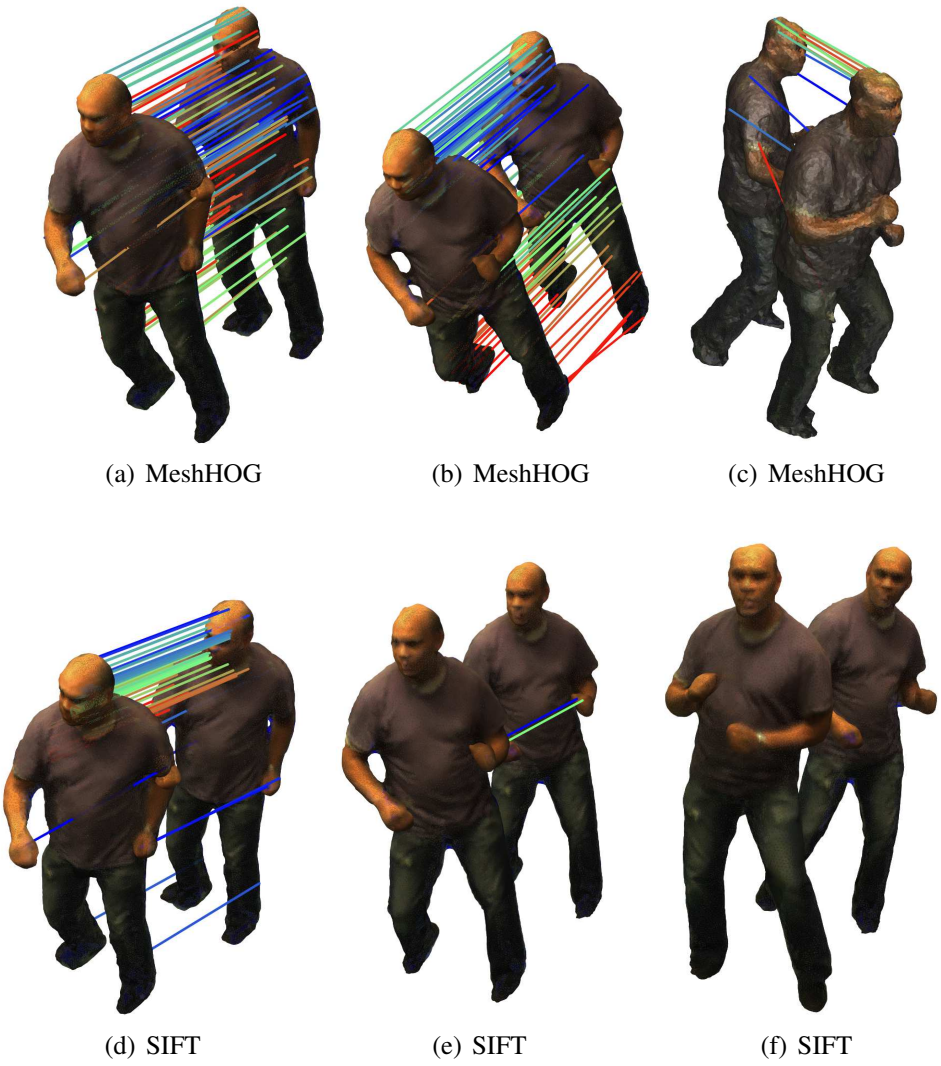
There are two kinds of uniformly distributed noise being applied: geometry noise (changing the vertices v) and colour noise (changing values $f(v)$ held in each vertex). The colour noise relates to % of the total amount of a maximum 255



(a) MeshHOG (b) MeshHOG (c) MeshHOG

(d) SIFT (e) SIFT (f) SIFT

Figure 7.11: Non Rigid matching using synthetic data - *dancer-synth* dataset. Comparison between MeshHOG and SIFT matching results. Matches between frames 1 and 50 are visually depicted in (a),(d). There are 364 matches for MeshHOG and 343 matches for SIFT. They are also quantified in the error histograms (b),(e). The histogram bins are of size equal to e_{avg}. The last bin groups all the errors greater than $20 * e_{avg}$. Additionally, the average histogram errors are shown in (c),(f) for matching frame 1 with x, where $x \in [2..200]$.



(a) MeshHOG (b) MeshHOG (c) MeshHOG

(d) SIFT (e) SIFT (f) SIFT

Figure 7.12: Non Rigid matching using real data - *Dance-1* sequence. Comparison between MeshHOG and SIFT matching results. Matches between frames 525 and 526 are visually depicted in (a), (d). There are 119 matches for MeshHOG and 54 matches for SIFT. Matches between frames 530 and 531 are visually depicted in (b), (e). There are 122 matches for MeshHOG and 2 matches for SIFT. Matches between frames 530 and 550 are visually depicted in (c), (f). There are 13 matches for MeshHOG and 0 matches for SIFT.

RGB value noise, whereas the geometry noise relates to the % of the total amount of a maximum e_{avg} noise level. As it can be observed in Figure 7.13, the method does not generate more false positives when the amount of noise increases. The *Dino* dataset has a larger number of false positives, since the two meshes are not perfectly identical, being the result of a 3-D reconstruction method from multiple images, which introduces some errors. In the *Synth-dance* dataset, the colour noise influences the descriptor accuracy more than the geometry noise, whereas in the *Dino* dataset the situation is reversed. This stems from the fact that the meshes in the two datasets have a relatively different number of vertices, which will in turn directly influence the ring neighbourhood size r ($r = 7$ for *Synth-dance*, and $r = 15$ and $r = 16$ for *Dino*), always chosen to represent α_r of the total mesh area.

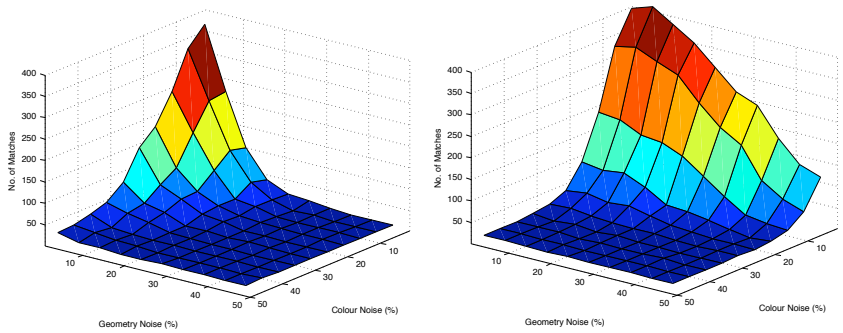
The running time of computing such a descriptor depends on the descriptor neighbourhood size. For example, in the *synth-dance* dataset, computing 706 descriptors using a neighbourhood size $r = 7$ took under 1 second, while computing 2724 descriptors using a ring neighbourhood size $r = 15$ took 35 seconds. The machine used for the test was a Core2Duo 2.4GHz Intel with 2 Gigs or RAM running Mac OS.X. The code has been developed in C++.

7.6.4 Integration with a Mesh Tracking Framework

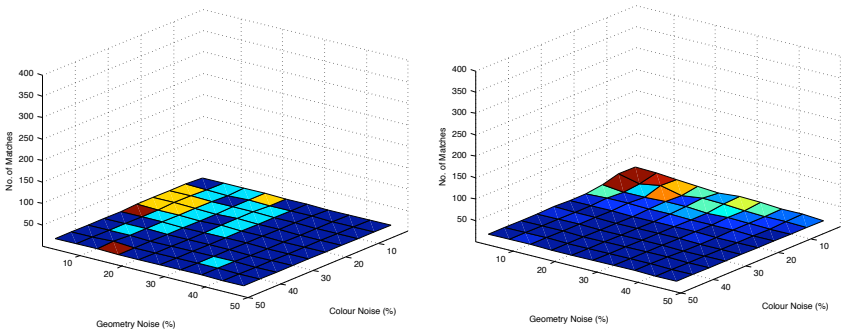
We have integrated the MeshHOG descriptor within an existing mesh tracking approach, described in [160]. The approach has three stages: finding a set of initial sparse matching, densifying the matches and mesh morphing. The set of initial matches consists of two steps: at first, the extremities are located and matched using the maxima of the geodesic integral and colour information. Secondly, new sparse correspondences are established using back-projected SIFT descriptors, together with additional geometric (elastic stretch and twist) and protrusion information. The sparse matches are diffused across, using laplacian processing. Lastly, mesh morphing is employed in order to handle potential topological changes and to guarantee convergence. The currently proposed MeshHOG descriptor is used in order to provide the set of initial sparse matches, replacing the back-projected SIFT descriptors. Some results are presented in Figure 7.14.

7.7 Conclusion

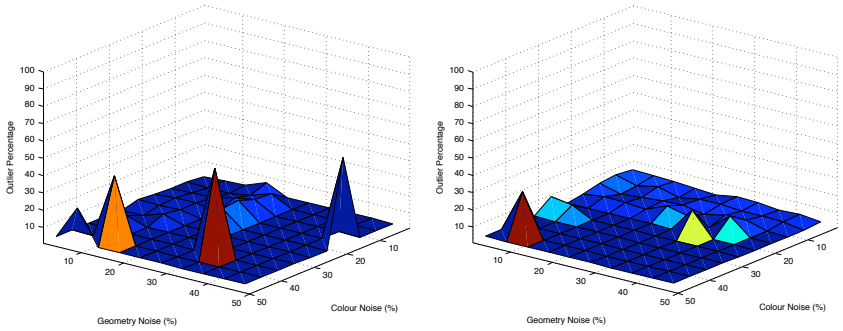
We have introduced MeshDOG and MeshHOG, a new 3-D interest point detector and a new 3-D descriptor, defined on uniformly sampled triangular meshes. The



(a) Synth - TP (b) Dino - TP



(c) Synth - FP (d) Dino - FP



(e) Synth - FP Ratio (f) Dino - FP Ratio

Figure 7.13: Resilience to noise measurements. There are two kinds of noise being applied: geometry noise (changing the vertices v) and colour noise (changing values $f(v)$ held in each vertex). Results are shown for the *Synth-dance* dataset (frame 1 and 50) in (a), (c), (e) and for the *Dino* dataset in (b), (d) (f). Graphics are presented for the True Positives (TP), the False Positives (FP) and the False Positive Ratio.

(a) (b)

Figure 7.14: INRIA Dance-1 Sequence: Frames 515 - 530. (a) Sparse Matching Results; (b) Final Tracking Results.

descriptor is able to capture the local geometric and/or photometric properties in a succinct fashion. It is robust to changes in orientation, rotation, translation and scale. We have presented results of matching various rigid and non rigid datasets, both on real sequences and on synthetically generated data. They demonstrate that local features detected on meshes using both photometric and geometric information are more robust than traditional purely photometric features detected in images. Lastly, we have shown how the above mentioned approach integrates within a mesh tracking framework.

Chapter 8

Conclusion

8.1 Summary

In this thesis we have addressed the necessary steps necessary to build a mesh-tracking framework using multiple camera environments: camera calibration and sparse 3-D reconstruction, dense 3-D reconstruction, sparse and dense mesh matching.

Firstly, we have developed a bayesian framework that works in conjunction with any affine factorization algorithm, able to recover both the extrinsic parameters of multiple cameras and 3-D coordinates of control points, given their projected 2-D point correspondences and the intrinsic camera parameters. The proposed framework is robust to outliers and compares favourably with bundle adjustment, a standard non-linear minimization technique, which requires an initial solution not very far from the optimum.

Secondly, we have proposed TransforMesh, a provably correct mesh-based surface evolution approach, able to handle topological changes and self-intersections without imposing any mesh sampling constraints. The exact mesh geometry is preserved throughout, except for the self-intersection areas. Sample applications, including mesh morphing and 3-D reconstruction using variational methods, are presented. Numerous examples are provided, proving its robustness. The link thus between sparse and dense 3-D reconstruction methods is established.

Thirdly, we have developed a scene-aware camera clustering method, able to break large scale reconstruction tasks in smaller independent partial reconstructions that are memory tractable. The method works by building a visibility matrix of feature points projections in the available cameras, thus being applicable either

when a rough geometry of the scene is available or key-point matches. By performing clustering on the transpose of the visibility matrix, regions of the scene can be clustered, instead of the cameras, which can be a very useful tool to generate virtual cameras in the scenarios where few high resolution images are available.

Lastly, we proposed a new 3 dimensional descriptor, entitled MeshHOG, defined on uniformly sampled triangular meshes, which is invariant to rotation, translation, scale, being able to capture local geometric and photometric properties. It is particularly useful in the multi-camera environments, where the reconstructed meshes benefit from colour/texture information. Nevertheless, the descriptor is defined generically for any feature available throughout the manifold, colour and curvatures being just some examples. Results in both rigid and non rigid matching tasks are presented. Additionally, the descriptor was integrated within a mesh tracking framework, providing dense matches.

The contributions follow intuitively the necessary steps required for a desired mesh tracking framework. Looking in retrospect, I think that the most valuable contribution that this thesis proposes is TransforMesh, which provides a provably correct method for triangular mesh surface evolution that does not suffer from sampling issues. I also consider the newly proposed mesh descriptor a valuable tool, applicable in a variety of mesh matching scenarios.

8.2 Future Work

The first two contributions (robust factorization and the surface evolution methods) have been thoroughly explored throughout this thesis and we do not envision any extensions in the near future. Nevertheless, the camera clustering and the 3-D mesh descriptor are both areas that can benefit from further enhancements. We will detail each of the proposed future contributions below.

8.2.1 Camera Clustering

In terms of camera clusterings, one area of future research is how to properly join together efficiently partial reconstructions obtained using separate camera subclusters. In the current implementation of the content aware clustering algorithm, one 3-D representation is used, which is subsequently minimized by each participating cluster. As a consequence, one cannot benefit from parallelism directly, not being able to run the minimization process on the camera clusters in parallel. While a number of methods [20, 12, 83, 90, 8, 128] able to reconstruct surface

representations from oriented point sets already exist and have been discussed in the introduction section, this issue will nevertheless pose problems when dealing with regions that suffer from not-sufficient samplings. Additionally, one should take into account the local mesh connectivity provided by each partial reconstruction. Given the fact that the proposed surface evolution technique works with open meshes, a possible solution is to run multiple minimization simultaneously by blocking the borders, followed by a merging step.

On a slightly different but related topic, the whole SLAM (Simultaneous Localization and Mapping) framework can benefit from some improvements that take camera clustering into account. In SLAM, a typical scenario is to have one camera video stream used in order to progressively cover a scene. The cameras and a sparse world representation are gradually recovered. They are to be integrated with an already dense representation of the world and cameras. When incorporating the new information, a camera clustering approach can dictate which existing cameras should be used in conjunction with the new ones in order to densely refine the world representation.

8.2.2 3-D Mesh Descriptor

There are a number of possible applications that can benefit from the proposed 3-D mesh descriptor. One possible application is to perform alignment of different partial reconstructions. On of the main drawbacks of SLAM like algorithms is the fact that over long sequences a certain amount of drift is accumulated, mainly due to the fact that only local minimizations are taken into account. Let us consider a scenario where one surrounds a building filming it. When recovering the 3-D scene using a SLAM like approach, the start and end point will not coincide. This could be corrected if, based on 3-D mesh descriptors, one detected that in fact the two scenes are the same, and therefore re-aligned them. Such an approach was proposed in [167] in conjunction with a different descriptor.

Alternatively, MeshHOG could be used as the building block in a 3-D object/category recognition framework, as it has been already done with other descriptors in [143], using the freely available Princeton Shape Benchmarking database [1]. The standard approach used in such scenarios is the so called *bag-of-words* approach [147]. Objects are characterized by a number of relevant descriptors, which are later on used for learning and classification. As future work, we plan on integrating our descriptor within the proposed framework and compare it with already existing approaches. There are over 12 shape descriptors already existent and currently implemented in [143].

[1] http://shape.cs.princeton.edu/benchmark/

One other possible extension of the current work would be to adapt the MSER (Maximally Stable Extremal Regions) interest point detector [108] to triangular meshes. As an advantage, this would also provide the neighbourhood size for the descriptor, which is otherwise selected based on a percentage of the total surface area.

Appendix A

Appendix: Factorization

A.1 Derivation of equation (4.19)

We recall eq. (4.17):

$$Q_{ML} = \frac{1}{2}\sum_{i,j}\Big((\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-1}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})) + \log(\det \mathbf{C})\Big)$$

Taking the derivative with respect to the entries of the 2×2 matrix $\mathbf{C}$ we obtain:

$$\frac{\partial Q_{ML}}{\partial \mathbf{C}} = \frac{1}{2}\sum_{i,j}\Big(- \mathbf{C}^{-\top}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top \mathbf{C}^{-\top} + \mathbf{C}^{-\top}\Big) \tag{A.1}$$

$$= -\mathbf{C}^{-\top}\Big(\frac{1}{2}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top\Big)\mathbf{C}^{-\top} + \frac{m}{2}\mathbf{C}^{-\top} \tag{A.2}$$

where $m = kn$. By setting the derivative to zero we obtain eq. (4.19):

$$\mathbf{C} = \frac{1}{m}\sum_{i,j}(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))(\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}))^\top$$

A.2 Derivation of equation (4.21)

When considering isotropic covariance, $\mathbf{C} = \sigma^2\mathbf{I}$, hence $\det \mathbf{C} = \sigma^4$, and the equation becomes:

$$Q_{ML} = \frac{1}{2}\sum_{i,j}\left(\frac{1}{\sigma^2}\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})\|^2 + 2\log(\sigma^2)\right) \tag{A.3}$$

By taking the derivative with respect to σ^2, we obtain:

$$\frac{\partial Q_{ML}}{\partial \sigma^2} = \frac{1}{2}\sum_{i,j}\left(-\frac{1}{(\sigma^2)^2}\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})\|^2 + 2\frac{1}{\sigma^2}\right) \tag{A.4}$$

$$= \frac{1}{2}\sum_{i,j}\left(\frac{2\sigma^2 - \|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta})\|^2}{\sigma^2}\right) \tag{A.5}$$

By setting the derivative to zero, we obtain eq. (4.21):

$$\sigma^2 = \frac{1}{2m}\sum_{i,j}\|\boldsymbol{s}_{ij} - \hat{\boldsymbol{s}}_{ij}(\boldsymbol{\theta}^*))\|^2$$

where $m = kn$.

A.3 Affine Power Factorization with Uncertainty

In the affine factorization step, λ_{ij} and w_{ij} are considered known. We also recall the definition $\boldsymbol{s}_{ij}^{weak} = (\epsilon_{ij} + 1)\boldsymbol{s}_{ij}$. In addition, we introduce the variables $w_{ij} = \mu_{ij}\alpha_{ij}^{in}$, combining the inlier information. Thus, the affine factorization problem to be solved becomes:

$$\min_{\mathbf{M},\mathbf{P}}\sum_{ij}\|w_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\|^2_{\Sigma_{ij}} \tag{A.6}$$

In order to solve the minimization problem stated in (A.6), we will employ a variant of the Power Factorization, namely Rank-4 Power Factorization with Uncertainty [68], which solves the following:

$$\min_{\mathbf{M},\mathbf{P}}\sum_{ij}\|\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\|^2_{\Sigma_{w_{ij}}} \tag{A.7}$$

We define:

$$\Sigma_{w_{ij}} = \frac{1}{w_{ij}^2}\Sigma_{ij} \tag{A.8}$$

and show how eq. (A.6) can be reduced to eq. (A.7):

$$\begin{aligned}
F &= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|w_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\|^2_{\Sigma_{ij}} \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[w_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\right]^T \Sigma_{ij}^{-1} \left[w_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right]^T w_{ij}^2\Sigma_{ij}^{-1} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right]^T \left[\frac{1}{w_{ij}^2}\Sigma_{ij}\right]^{-1} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right]^T \mathbf{\Sigma}_{w_{ij}}^{-1} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\|^2_{\Sigma_{w_{ij}}}
\end{aligned} \tag{A.9}$$

The Method

Below we will present the solution to the PowerFactorization with Uncertainty algorithm presented in [68], as stated in eq. (A.7). We believe that it is important to include the solution here, in order to provide the reader with a clear complete understanding of the overall technique.

The solution of the PowerFactorization method falls within the alternation methods, which solve linearly for one of the matrices $\mathbf{M}$ or $\mathbf{P}$, considering the other matrix known. The algorithm iterates until convergence (when the values do not change anymore). An attractive feature of the PowerFactorization method is that it converges in few iterations, starting from a random matrix.

We have $\Sigma_{w_{ij}}^{-1} = \mathbf{V}_{ij}^T\mathbf{V}_{ij}$, since $\Sigma_{w_{ij}}^{-1}$ is symmetric and positive semi-definite. Thus:

$$\begin{aligned}
F &= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\|^2_{\Sigma_{w_{ij}}} \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right]^T \mathbf{\Sigma}_{w_{ij}}^{-1} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right]^T \mathbf{V}_{ij}^T\mathbf{V}_{ij} \left[\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j\right] \\
&= \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\mathbf{V}_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\|^2_E
\end{aligned} \tag{A.10}$$

Solving for the $\mathbf{M}_i$. Assume that all the points $\mathbf{P}_j$ are known, and the following notations:

$$\overset{2\times4}{\mathbf{M}_i} \times \overset{4\times1}{\mathbf{P}_j} = \begin{pmatrix} \boldsymbol{m}_i^1 \\ \boldsymbol{m}_i^2 \end{pmatrix} P_j = \underbrace{\begin{bmatrix} \overset{2\times8}{\mathbf{P}_j^T} & 0^T \\ 0^T & \mathbf{P}_j^T \end{bmatrix}}_{\bar{\mathbf{P}}_j} \times \underbrace{\begin{pmatrix} \overset{8\times1}{\boldsymbol{m}_i^{1T}} \\ \boldsymbol{m}_i^{2T} \end{pmatrix}}_{\bar{\boldsymbol{m}}_i} = \bar{\mathbf{P}}_j \bar{\boldsymbol{m}}_i \quad \text{(A.11)}$$

With this notation, the function becomes:

$$F = \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\mathbf{V}_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\|_E^2 = \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\mathbf{V}_{ij}\boldsymbol{s}_{ij}^{weak} - \mathbf{V}_{ij}\bar{\mathbf{P}}_j\bar{\boldsymbol{m}}_i)\|_E^2$$

which is minimized by solving the *normal equations*, leading to:

$$\bar{\boldsymbol{m}}_i = \left(\sum_j \bar{\mathbf{P}}_j^T \Sigma_{w_{ij}}^{-1} \bar{\mathbf{P}}_j \right)^{-1} \left(\sum j \bar{\mathbf{P}}_j^T \Sigma_{w_{ij}}^{-1} \boldsymbol{s}_{ij}^{weak} \right) \quad \text{(A.12)}$$

Solving for the $\mathbf{P}_j$. Assume that all the points $\mathbf{M}_i$ are known, and the following notations:

$$\overset{2\times4}{\mathbf{M}_i} \times \overset{4\times1}{\mathbf{P}_j} = \begin{pmatrix} \overset{2\times3}{\tilde{\mathbf{M}}_i} & \overset{2\times1}{\boldsymbol{t}_i} \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{P}}_j \\ 1 \end{pmatrix} \quad \text{(A.13)}$$

With this notation, the function becomes:

$$F = \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\mathbf{V}_{ij}(\boldsymbol{s}_{ij}^{weak} - \mathbf{M}_i\mathbf{P}_j)\|_E^2 = \min_{\mathbf{M},\mathbf{P}} \sum_{ij} \|\mathbf{V}_{ij}(\boldsymbol{s}_{ij}^{weak} - \boldsymbol{t}_i) - \mathbf{V}_{ij}\tilde{\mathbf{P}}_j\tilde{\boldsymbol{m}}_i)\|_E^2$$

which is minimized by solving the *normal equations*, leading to:

$$\tilde{\mathbf{P}_j} = \left(\sum_i \tilde{\mathbf{M}}_i^T \Sigma_{w_{ij}}^{-1} \tilde{\mathbf{M}}_i \right)^{-1} \left[\sum_i \tilde{\mathbf{M}}_i^T \Sigma_{w_{ij}}^{-1} (\boldsymbol{s}_{ij}^{weak} - t_i) \right] \quad \text{(A.14)}$$

Appendix B

Full Publication List

Below the reader will find listed all the publications that sprung out of the work presented in the current manuscript:

Papers that appeared in Conference Proceedings:

- A. Zaharescu, R. Horaud, R. Ronfard, and L. Lefort. Multiple camera calibration using robust perspective factorization. In *Proceedings of International Symposium on 3D Data Processing, Visualisation and Transmission*, 2006. [173]
- A. Zaharescu, E. Boyer, and R. P. Horaud. Transformesh: a topology- adaptive mesh-based approach to surface evolution. In *Proceedings of Asian Conference on Computer Vision*, 2007. [170]
- K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In *Proceedings of European Conference on Computer Vision*, 2008 [160]
- A. Zaharescu, C. Cagniart, S. Ilic, E. Boyer, and R. Horaud. Camera- clustering for multi-resolution 3-D surface reconstruction. In *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008 [172]

Journal Articles:

- A. Zaharescu and R. P. Horaud. Robust factorization methods using a gaussian/uniform mixture model. *International Journal of Computer Vision*, accepted, to appear in 2009 [174]

- A. Zaharescu, E. Boyer, and R. P. Horaud. A mesh-based topology-adaptive self-intersection removal algorithm for surface evolution. *IEEE Transactions on Visualization and Compute Graphics*, accepted subject to revisions [171]

Bibliography

[1] H. Aanæs, R. Fisker, and K. Astrom. Robust factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, Sept. 2002. 2.2.1, 4.1, 4.1, 4.3, 4.7

[2] D. Adalsteinsson and J. Senthian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, 1995. 2.2.2, 5.1.1

[3] M. Aftosmis, M. Berger, and J. Melton. Robust and efficient cartesian mesh generation for component-based geometry. In *AIAA Paper 97-0196.*, 1997. 5.1.2, 5.2.2

[4] E. Aganj, J.-P. Pons, F. Segonne, and R. Keriven. Spatio-temporal shape from silhouette using four-dimensional delaunay meshing. In *Proceedings of International Conference on Computer Vision*, 2007. 2.3.2

[5] A. Agrawal and A. Requicha. A paradigm for the robust design of algorithms for geometric modeling. *Computer Graphics Forum*, 13(3):33–44, 1994. 5.1.1

[6] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 7.2

[7] J. Allard, C. Menier, B. Raffin, E. Boyer, and F. Faure. Grimage: Markerless 3D interactions. In *Proceedings of SIGGRAPH*, 2007. 1.1

[8] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry Processing*, pages 39–48, 2007. 2.2.2, 2.2.2, 8.2.1

[9] S. M. S. amd C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. 2.2.2, 2.2.2, 5.4.2

[10] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of SIGGRAPH*, pages 415–421, 1998. 2.2.2

[11] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12:125–141, 2002. 2.2.2

[12] N. Amenta, S. Choi, and R. Kolluri. The power crust, unions of balls, and the medial axis transform,. *Computational Geometry: Theory and Applications*, 19(2-3):127–153, 2001. 2.2.2, 2.2.2, 3.3.3, 3.13, 5.4.2, 5.16, 8.2.1

[13] P. Anandan and M.Irani. Factorization with uncertainty. *International Journal of Computer Vision*, 49(2/3):101–116, 2002. 2.2.1, 3.2, 4.1

[14] D. Anguelov, P. Srinivasan, H.-C. Pang, D. Koller, S. Thrun, , and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Proceedings of Neural Information Processing Systems*, 2004. 2.3.2

[15] B. G. Baumgart. *Geometric Modeling for Computer Vision*. PhD thesis, Standford University, 1974. 5.1.1

[16] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *Proceedings of SIGGRAPH*, pages 185–194, 2001. 5.1.1

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2.4, 3.4.1, 4.1, 4.3, 6.2

[18] C. E. Board. *CGAL-3.2 User and Reference Manual*, 2006. 5.3.4, 5.3.8

[19] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984. 2.2.2

[20] J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the Symposium on Computational geometry*, pages 223–232, New York, NY, USA, 2000. ACM Press. 2.2.2, 2.2.2, 8.2.1

[21] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker - description of the algorithm. Technical report, Intel Corporation, 2001. 2.2.1, 3.2.4, 4.6

[22] I. C. Braid, R. C. Hillyard, and I. A. Stroud. Stepwise construction of polyhedra in geometric modelling. *Mathematical Methods in Computer Graphics and Design*, 1978. 5.1.1

[23] S. Brant. Closed-form solutions for affine reconstruction under missing data. In *Proceedings of Statistical Methods for Video Processing (ECCV '02 Workshop)*, pages 109–114, 2002. 4.1

[24] D. E. Breen and R. T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transaction on Visualization and Computer Graphics*, 7(2):173–192, 2001. 3.3.3, 5.4.1

[25] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, to appear, 2008. 2.3.1

[26] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Calculus of non-rigid surfaces for geometry and texture manipulation. *International Journal of Computer Vision*, 13(5):902–913, 2007. 2.3.2

[27] M. Brown and D. G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proceedings of Conference on 3D Imaging and Modelling*, pages 56–63, 2005. 2.2.1, 2.2.1

[28] A. M. Buchanan. Investigation into matrix factorization when elements are unknown. Technical report, University of Oxford, University of Oxford, `http://www.robots.ox.ac.uk/~amb`, 2004. 1.2

[29] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 316–322, 2005. 2.2.1, 4.7

[30] B. Bustos, D. A. Keim, D. Saupe, T. Schreck, and D. V. Vranic. Feature-based similarity search in 3D object databases. *ACM Computing Surveys*, 34(4):345–387, 2005. 2.3.1, 7.2

[31] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *Proceedings of SIGGRAPH*, pages 569–577, July 2003. 2.3.2

[32] S. Christy and R. Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, November 1996. 2.2.1, 3.2.3, 4.1, 4.4, 4.4

[33] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005. 2.4, 3.5.1, 7.1, 7.2

[34] E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Marker-less 3D feature tracking for mesh-based human motion capture. In *Human Motion – Understanding, Modeling, Capture and Animation*, pages 1–15, 2007. 1.1, 2.3.2, 7.2

[35] D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, 2000. 2.3.2

[36] H. Delingette, M. Herbert, and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and Vision Computing*, pages 132–145, 1992. 3.3.2, 5.4

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977. 3.2, 4.1, 4.3

[38] M. Desbrun, M. Meyer, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH*, 1999. 2.2.2

[39] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: a comparison study for a voronoi based method. In *Eurographics Symposium on Point-Based Graphics*, pages 39–46, 2005. 2.2.2

[40] C.-S. Dong and G. Z. Wang. Curvatures estimation on triangular mesh. *Journal of Zhejiang University SCIENCE*, 6A(1):128–136, 2005. 2.2.2

[41] Y. Duan, L. Yang, H. Qin, and D. Samara. Shape reconstruction from 3D and 2D data using pde-based deformable surfaces. In *Proceedings of European Conference on Computer Vision*, volume 3, pages 238–251, 2004. 5.1.1, 6.1

[42] H. Edelsbrunner and E. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990. 5.3.4

[43] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002. 5.1.1

[44] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of SIGGRAPH*, pages 736–744, 2002. 5.1.1

[45] O. Faugeras. *Three Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993. 2.1.3, 2.2.1, 4.5

[46] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. 1.2, 2.2.1, 2.2.1, 4.1

[47] J. D. Foley, A. van Dam, S. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison Wesley, 1990. 5.1.1

[48] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002. 3.2, 4.1, 4.3

[49] J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 329–338, September 2003. 1.1, 2.2.2, 3.3.3, 5.4.2, 5.4.2

[50] J.-S. Franco and E. Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. In *Proceedings of International Conference on Computer Vision*, pages 1747–1753, 2005. 2.2.2

[51] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of European Conference on Computer Vision*, May 2004. 2.3.1, 7.2

[52] P. Fua. From multiple stereo views to multiple 3-D surfaces. *International Journal of Computer Vision*, 24(1):19–35, 1997. 2.2.2

[53] Y. Furukawa and J. Ponce. High-fidelity image-based modeling. Technical Report 02, University of Illinois at Urbana-Champaign, 2006. 2.2.1

[54] Y. Furukawa and J. Ponce. Accurate, dense and robust multi-view stereopsis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 2.2.1, 2.2.2, 2.2.2, 3.3.3, 3.3.3, 5.4.1, 5.4.2, 6.1

[55] Y. Furukawa and J. Ponce. Dense 3D motion capture from synchronized video streams. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 7.2

[56] P. Gargallo. *Contributions to the Bayesian Approach to Multi-View Stereo*. PhD thesis, Grenoble Institute of Technology, 2008. 2.2.2

[57] P. Gargallo and P. Sturm. Bayesian 3D modeling from images using multiple depth maps. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 885–891, 2005. 2.2.2

[58] P. Gargallo, P. Sturm, and S. Pujades. An occupancy-depth generative model of multi-view images. In *Proceedings of Asian Conference on Computer Vision*, volume 2, pages 373–383, 2007. 2.2.2

[59] D. Gavrila and L. Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1996. 1.1, 2.3.2

[60] J. Goldfeather, J. P. M. Hultquist, and H. Fuchs. Fast constructive-solid geometry display in the pixel-powers graphics system. In *Proceedings of SIGGRAPH*, volume 20, pages 107–116, 7 1986. 5.1.1

[61] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989. 4.1

[62] A. Gruber and Y. Weiss. Factorization with uncertainty and missing data: exploring temporal coherence. In *Proceedings of Neural Information Processing Systems*, 2003. 2.2.1, 4.1

[63] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 707–714, 2004. 4.1

[64] A. Gueziec, G. Taubin, F. Lazarus, and B. Horn. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transaction on Visualization and Computer Graphics*, 7(2):136–151, 2001. 5.2.3

[65] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 6.1

[66] E. Haines. *Graphic Gems IV*, chapter Point in Polygon Strategies, pages 24–46. Academic Press, 1994. 5.2.2

[67] L. Hajder and D. Chetverikov. Robust structure from motion under weak perspective. In *Proceedings of International Symposium on 3D Data Processing, Visualisation and Transmission*, September 2004. 4.1

[68] R. Hartley and F. Schaffalitzky. Powerfactorization: 3D reconstruction with missing or uncertain data. Technical report, Australian National University, 2003. 3.2, 3.2.2, 4.1, 4.3, 4.5, 4.7, A.3, A.3

[69] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000. 2.1.2, 2.1.3, 2.2.1, 2.2.1, 4.5, 4.5

[70] C. E. Hernández and F. Schmitt. Silhouette and stereo fusion for 3-D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004. 3.3.3, 3.3.3, 5.1.1, 5.4.2

[71] S. Hert and M. Seel. dD convex hulls and delaunay triangulations. In C. E. Board, editor, *CGAL-3.2 User and Reference Manual*. 2006. 5.2.2

[72] G. R. Hjaltason and H. Samet. Ranking in spatial databases. *Symposium on Large Spatial Databases*, pages 83–95, 1995. 5.4.1

[73] H. Hoppe, T. DeRose, T. Duchamp, J. MCDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH*, 1992. 5.4.1

[74] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer. A.*, 4(4):629–642, 1987. 4.5

[75] P. M. Hubbard. Constructive solid geometry for triangulated polyhedra. Technical Report CS-90-07, Department of Computer Science, Brown University, 1 1990. 5.1.1

[76] D. Q. Huynh, R. Hartley, and A. Heyden. Outlier correction in image sequences for the affine camera. In *Proceedings of International Conference on Computer Vision*, volume 1, pages 585–590, 2003. 4.1

[77] D. Q. Huynh and A. Heyden. Robust factorization for the affine camera: Analysis and comparison. In *Proc. Seventh Interanational Conference on Control, Automation, Robotics, and Vision*, Singapore, December 2002. 4.1

[78] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999. 2.3.1, 7.2

[79] M. W. Jones, J. A. Bærentzen, and M. Sramek. 3D distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Compute Graphics*, 12(4):581–599, July/August 2006. 5.4.1

[80] W. Jung, H. Shin, and B. K. Choi. Self-intersection removal in triangular mesh offsetting. *Computer-Aided Design and Applications*, 1(1-4):477–484, 2004. 5.1.2, 6.1

[81] F. Kahl. Multiple view geometry and the L_∞-norm. In *Proceedings of International Conference on Computer Vision*, 2005. 2.2.1

[82] I. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, december 2000. 1.1, 2.3.2

[83] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70, 2006. 2.2.2, 2.2.2, 6.2, 8.2.1

[84] L. Kettner, A. Meyer, and A. Zomorodian. Intersecting sequences of dD iso-oriented boxes. In C. E. Board, editor, *CGAL-3.2 User and Reference Manual*. 2006. 5.2.1

[85] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *Proceedings of the British Machine Vision Conference*, 2008. 7.2, 7.5

[86] D. Knossow, R. Ronfard, and R. P. Horaud. Human motion tracking with a kinematic parameterization of extremal contours. *International Journal of Computer Vision*, 79(2):247–269, September 2008. 1.1

[87] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Proceedings of Eurographics*, pages 249–260, 2000. 3.3.2, 5.4, 7.3

[88] M. Körtgen, G.-J. Park, M. Novotny, and R. Klein. 3D shape matching with 3D shape contexts. *Central European Seminar on Computer Graphics*, 2003. 2.3.1, 7.2

[89] K. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000. 2.2.2

[90] 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 *Proceedings of International Conference on Computer Vision*, 2007. 2.2.2, 2.2.2, 8.2.1

[91] J.-O. Lachaud and B. Taton. Deformable model with adaptive mesh and automated topology changes. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling*, 2003. 2.2.2, 5.1.1, 6.1

[92] S. Lazebnik, Y. Furukawa, and J. Ponce. Projective visual hulls. *International Journal of Computer Vision*, 74(2):137–165, 2007. 2.2.2

[93] P. Lindstrom and G. Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Compute Graphics*, 5(2):98–115, 1999. 7.4

[94] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. 2.2.2, 5.1.1

[95] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35(10):995–1010, 2006. 5.1.1

[96] M. I. A. Lourakis and A. A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report FORTH-ICS TR-340-2004, Institute of Computer Science - FORTH, August 2004. 4.7

[97] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 2.2.1, 3.5.1, 3.4.1, 7.1, 7.2, 7.4

[98] Q. Luong and O. Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *International Journal of Computer Vision*, 1:5–40, 1997. 2.1.3, 4.5

[99] Q.-T. Luong and O. D. Faugeras. *The Geometry of Multiple Images*. MIT Press, Boston, 2001. 2.1.3, 4.5

[100] S. Mahamud and M. Hebert. Iterative projective reconstruction from multiple views. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 430 – 437, June 2000. 2.2.1, 4.1, 4.4

[101] S. Mahamud, M. Hebert, Y. Omori, and J. Ponce. Provably-convergent iterative methods for projective structure from motion. In *Proceedings of IEEE*

Conference on Computer Vision and Pattern Recognition, 2001. 2.2.1, 4.1, 4.4

[102] D. Marr and E. Hildreth. Theory of edge detection. In *Proc. Roy. Soc. London*, volume B 207, pages 187–217, 1980. 7.1

[103] D. Martinec. *Robust Multiview Reconstruction*. PhD thesis, Czech Technical Univeristy in Prague, 2008. 2.2.1

[104] D. Martinec and T. Pajdla. Structure from many perspective images with occlusions. In *Proceedings of European Conference on Computer Vision*, volume 2, pages 355–369, 2002. 2.2.1

[105] D. Martinec and T. Pajdla. Consistent multi-view reconstruction from epipolar geometries with outliers. In *Proceedings of Scandinavian Conference on Image Analysis*, pages 493–500, 2003. 2.2.1

[106] D. Martinec and T. Pajdla. 3d reconstruction by fitting low-rank matrices with missing data. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–205, 2005. 2.2.1

[107] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 2.2.1

[108] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, 2002. 2.2.1, 8.2.2

[109] T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4(2):73–91, 2000. 2.2.2, 5.1.1, 6.1

[110] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New-York, 1997. 3.2, 4.1, 4.3

[111] P. Meer. Robust techniques for computer vision. In *Emerging Topics in Computer Vision*. Prentice Hall, 2004. 2.2.1, 4.1, 4.1

[112] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nister, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *Proceedings of International Conference on Computer Vision*, 2007. 2.2.2, 2.2.2, 6.1

[113] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential geometry operators for triangulated 2-dimensional manifolds. In *Proceedings of VisMath*, 2002. 3.3.2, 5.4

[114] K. Mikolajczyk and C. Schmidt. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. 7.2

[115] D. J. Miller and J. Browning. A mixture model and em-based algorithm for class discovery, robust classification and outlier rejection in mixed labeled/unlabeled data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1468–1483, November 2003. 4.1

[116] I. Miyagawa and K. Arakawa. Motion and shape recovery based on iterative stabilization for modest deviation from planar motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1176–1181, July 2006. 4.1, 4.4

[117] D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Proceedings of International Conference on Computer Vision*, pages 696–702, 1998. 3.2, 4.1

[118] A. L. Nathan Litke and P. Schröder. Trimming for subdivision surfaces. Technical report, Caltech, 2000. 5.1.1

[119] J. Neumann and Y. Aloimonos. Spatio-Temporal Stereo Using Multi-Resolution Subdivision Surfaces. *International Journal of Computer Vision*, 47:181–193, 2002. 2.3.2

[120] J. Novatnack and K. Nishino. Scale-dependent 3D geometric features. In *Proceedings of International Conference on Computer Vision*, 2007. 7.2

[121] J. Novatnack and K. Nishino. Scale-dependent/invariant local 3D shape descriptors for fully automatic registration of multiple sets of range images. In *Proceedings of European Conference on Computer Vision*, 2008. 7.2

[122] J. Oliensis and R. Hartley. Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2217–2233, December 2007. 2.2.1, 4.1, 4.4

[123] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003. 2.2.2, 5.1.1, 5.4.1, 6.1

[124] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer, 2003. 5.1.1

[125] S. Osher and J. Senthian. Front propagating with curvature dependent speed: algorithms based on the Hamilton-Jacobi formulation. *Journal of computational Physics*, 79(1):12–49, 1988. 2.2.2, 5.1.1

[126] J.-J. Park, T. McInerney, D. Terzopoulos, and M.-H. Kim. A non-self-intersection adaptive deformable surface for complex doundary extraction from volumetric images. *Computer & Graphics*, 25:421–440, 2001. 5.1.1

[127] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *Proceedings of SIGGRAPH*, pages 641–650, 2003. 2.2.2

[128] S. Petitjean and E. Boyer. Regular and non-regular point sets: Properties and reconstruction. *Computational Geometry-Theory and Application*, 19(2-3):101–126, 2001. 2.2.2, 2.2.2, 8.2.1

[129] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 3(59):207–232, 2004. 2.2.1, 2.2.1, 2.2.2

[130] J.-P. Pons and J.-D. Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, USA, Jun 2007. 2.2.2, 5.1.1, 6.1

[131] J.-P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179 – 193, 2007. 2.2.2, 2.3.2, 2.4, 3.3.3, 3.3.3, 5.4.2, 5.4.2, 5.4.2, 5.4.2, 6.1, 6.2

[132] S. Ramalingam. *Generic Imaging Models: Calibration and 3D Reconstruction Algorithms*. PhD thesis, Grenoble Institute of Technology, 2006. 2.1

[133] A. Rappoport and S. Spitz. Interactive boolean operations for conceptual design of 3-D solids. In *Proceedings of SIGGRAPH*, pages 269–278, 1997. 5.1.1

[134] J. Rossignac and A. Requicha. *Encyclopedia of Electrical and Electronics Engineering*, chapter Solid Modeling. John Wiley and Sons, 1999. 5.1.1

[135] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984. 4.1

[136] P. J. Rousseeuw and S. Van Aelst. Positive-breakdown robust methods in computer vision. In Berk and Pourahmadi, editors, *Computing Science and Statistics*, volume 31, pages 451–460. Interface Foundation of North America, 1999. 4.1

[137] S. Roweis. EM algorithm for PCA and SPCA. *Proceedings of Neural Information Processing Systems*, 10:626–632, 1997. 4.1

[138] M. Salzmann, J.Pilet, S.Ilic, and P.Fua. Surface deformation models for non-rigid 3–d shape recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1481–1487, 2007. 2.3.2

[139] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002. 2.2.1, 2.2.2

[140] M. Schlattmann, P. Degener, and R. Klein. Scale space based feature point detection on surfaces. *Journal of WSCG*, 16(1-3), February 2008. 7.2

[141] D. Schreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*. Addison-Wesley, 5 edition, 2006. 6.2

[142] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–526, 2006. 1.1, 1.2, 2.2.2, 3.3.3, 3.4.2, 5.1.2, 5.4.2, 6.3.2, 7.1, 7.6

[143] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. In *Shape Modeling International*, 2008. 7.2, 8.2.2

[144] H. Shin, J. C. Park, B. K. Choi, Y. C. Chung, and S. Rhee. Efficient topology construction from triangle soup. In *Proceedings of the Geometric Modeling and PRocessing*, 2004. 5.2.3

[145] H. Shum, K.Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):855–867, 1995. 4.1

[146] I. Simon, N. Snavely, and S. Seitz. Scene summarization for online image collections. In *Proceedings of International Conference on Computer Vision*, pages 1–8, 2007. 3.4.1, 6.1, 6.2

[147] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering object categories in image collections. In *Proceedings of International Conference on Computer Vision*, 2005. 8.2.2

[148] J. Starck and A. Hilton. Correspondence labelling for wide-time free-form surface matching. In *Proceedings of International Conference on Computer Vision*, 2007. 2.3.2, 7.2, 7.6

[149] J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007. 7.4

[150] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537, 1999. 2.2.1, 4.1, 4.1, 4.3, 4.8

[151] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 6.3.1

[152] P. Sturm and W. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proceedings of European Conference on Computer Vision*, volume 1065 of *LNCS*, pages 709–720, April 1996. 2.2.1, 4.1, 4.4

[153] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994. 5.1.1

[154] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Proceedings Shape Modeling International*, pages 145–156, 2004. 2.3.1, 7.2

[155] J. P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy. Algorithms for batch matrix factorization with application to structure-from-motion. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007. 2.2.1, 4.1, 4.6, 4.7

[156] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992. 2.2.1, 3.2, 4.7

[157] M. Trajkovic and M. Hedley. Robust recursive structure and motion recovery under affine projection. In *Proceedings of the British Machine Vision Conference*, September 1997. 4.1

[158] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–375. LNCS, 2000. 1.2, 2.2.1

[159] T. Ueshiba and F. Tomita. A factorization method for projective and euclidean reconstruction from multiple perspective views via iterative depth estimation. In *Proceedings of European Conference on Computer Vision*, pages 296–310, 1998. 2.2.1

[160] K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In *Proceedings of European Conference on Computer Vision*, 2008. 1.1, 2.5, 2.3.2, 2.4, 3.5.2, 7.2, 7.6.4, B

[161] S. Vedula, P. Rander, R. Collins, and T. Kanade. Three-Dimensional Scene Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):474–480, 2005. 2.3.2

[162] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 310–316, 2004. 4.1

[163] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 2.2.2

[164] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-D location parameters using dual number quaternions. *CGVIP-Image Understanding*, 54(3):358–367, November 1991. 4.5

[165] T. Wiberg. Computation of principal components when data are missing. In *Proceedings Symposium of Computational Statistics*, pages 229–326, 1976. 4.1

[166] S.-F. Wong and R. Cipolla. Extracting Spatiotemporal Interest Points using Global Information. In *Proceedings of International Conference on Computer Vision*, 2007. 7.2

[167] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys. 3D model matching with viewpoint invariant patches (vips). In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008. 2.3.1, 7.2, 8.2.2

[168] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of International Symposium on 3D Data Processing, Visualisation and Transmission*, 2008. 2.2.2, 2.2.2

[169] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *Proceedings of International Conference on Computer Vision*, 2007. 2.2.2, 2.2.2

[170] A. Zaharescu, E. Boyer, and R. P. Horaud. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Proceedings of Asian Conference on Computer Vision*, volume II of *LNCS 4844*, pages 166–175, November 2007. 6.1, 6.2, 6.2, 6.3.2, 7.6, B

[171] A. Zaharescu, E. Boyer, and R. P. Horaud. A mesh-based topology-adaptive self-intersection removal algorithm for surface evolution. *IEEE Transactions on Visualization and Compute Graphics*, accepted with revisions. B

[172] A. Zaharescu, C. Cagniart, S. Ilic, E. Boyer, and R. Horaud. Camera-clustering for multi-resolution 3-D surface reconstruction. In *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008. B

[173] A. Zaharescu, R. Horaud, R. Ronfard, and L. Lefort. Multiple camera calibration using robust perspective factorization. In *Proceedings of International Symposium on 3D Data Processing, Visualisation and Transmission*. IEEE Computer Society Press, 2006. 3.2.3, 4.4, 4.5, B

[174] A. Zaharescu and R. P. Horaud. Robust factorization methods using a gaussian/uniform mixture model. *International Journal of Computer Vision*, 2009. B

[175] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. 2.1.3, 4.5

[176] Z. Zhang. Camera calibration with one-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):892–899, 2004. 2.1.3, 4.5

[177] A. Zomorodian and H. Edelsbrunner. Fast software for box intersection. *International Journal of Compational Geometry and Applications*, 12(1-2):143–172, 2002. 5.2.1