IMPLICIT MESHES: UNIFYING IMPLICIT AND EXPLICIT SURFACE REPRESENTATIONS FOR 3D RECONSTRUCTION AND TRACKING

THÈSE N^{O} 3243 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédias

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Slobodan ILIC

Electrical engineer in computer science and informatics, University of Nis, Serbie
et de nationalité serbe

acceptée sur proposition du jury:

Prof. P. Fua, directeur de thèse
Dr R. Horaud, rapporteur
Prof. D. Thalmann, rapporteur
Prof. T. Vetter, rapporteur

Lausanne, EPFL
2005

Abstract

This thesis proposes novel ways both to represent the static surfaces, and to parameterize their deformations. This can be used both by automated algorithms for efficient 3–D shape reconstruction, and by graphics designers for editing and animation.

Deformable 3–D models can be represented either as traditional explicit surfaces, such as triangulated meshes, or as implicit surfaces. Explicit surfaces are widely accepted because they are simple to deform and render, however fitting them involves minimizing a non-differentiable distance function. By contrast, implicit surfaces allow fitting by minimizing a differentiable algebraic distance, but they are harder to meaningfully deform and render. Here we propose a method that combines the strength of both representations to avoid their drawbacks, and in this way build robust surface representation, called *implicit mesh*, suitable for automated shape recovery from video sequences. This surface representation lets us automatically detect and exploit silhouette constraints in uncontrolled environments that may involve occlusions and changing or cluttered backgrounds, which limit the applicability of most silhouette based methods.

We advocate the use of *Dirichlet Free Form Deformation* (DFFD) as generic surface deformation technique that can be used to parameterize objects of arbitrary geometry defined as explicit meshes. It is based on the small set of control points and the generalized interpolant. Control points become model parameters and their change causes model's shape modification. Using such parameterization the problem dimensionality can be dramatically reduced, which is desirable property for most optimization algorithms, thus makes DFFD good tool for automated fitting.

Combining DFFD as a generic parameterization method for explicit surfaces and implicit meshes as a generic surface representation we obtained a powerfull tool for automated shape recovery from images. However, we also argue that any other avaliable surface parameterization can be used.

We demonstrate the applicability of our technique to 3–D reconstruction of the human upper-body including – face, neck and shoulders, and the human ear, from noisy stereo and silhouette data. We also reconstruct the shape of a high resolution human faces parametrized in terms of a Principal Component Analysis model from interest points and automatically detected silhouettes. Tracking of deformable objects using implicit meshes from silhouettes and interest points in monocular sequences is shown in following two examples: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations.

Résumé

Cette thèse propose de nouveaux outils génériques permettant de représenter des surfaces statiques et de paramétrer leur déformations. Ces outils peuvent être utilisés à la fois pour la reconstruction automatique de surfaces, et par un concepteur humain pour l'édition et l'animation de modèles graphiques.

Un modèle tridimensionnel déformable peut être représenté soit par une surface explicite classique tel qu'un maillage triangulé, soit par une surface implicite. Le rendu graphique et la déformation de surfaces explicites sont relativement faciles à réaliser, et celles-ci sont plus fréquemment utilisées. Néanmoins, les mettre en correspondance avec des données tel qu'un nuage de points 3D nécessiterait la minimisation d'une fonction de distance non différentiable. Les surfaces implicites, en revanche, permettent de définir facilement une distance algèbrique différentiable, même si leur rendu est plus difficile et leurs déformations moins intuitives. Nous proposons une méthode combinant les points forts de ces deux représentations en une représentation robuste de surfaces, que nous appelons *implicit mesh*, ou *maillage implicite*. Cette représentation nous permet alors de reconstruire des objets tri-dimensionnels à partir d'une séquence vidéo. Les informations de silhouette peuvent être prises en compte facilement pour contraindre cette reconstruction, même dans le cas d'un environnement non controlé, où l'arrière-plan peut être complexe et des occultations peuvent survenir.

Nous montrons également que les *Dirichlet Free Form Deformation* (DFFD) sont un bon outil pour déformer des objets de géométrie arbitraire définis par un maillage explicite. Les DFFD sont basées sur un petit ensemble de points de contrôle et une interpolation généralisée. Les points de contrôle peuvent être utilisés comme paramètres, et leur modification est répercutée sur la forme du modèle. Grâce à cette paramétrisation, la dimensionalité de notre problème est considérablement réduit, ce qui facilite l'optimisation, et fait des DFFD un très bon outil pour la reconstruction automatique.

En utilisant les DFFD pour la paramétrisation de surface explicite, et les implicit meshes comme représentation de surface générique, nous obtenons un outil puissant pour l'estimation automatique de formes à partir d'images.

Nous montrons que notre technique nous permet de reconstruire en 3D le haut du corps d'une personne, en particulier le visage, le cou et les épaules, et aussi la forme d'une oreille, à partir de données bruitées constituées de points 3D et de silhouettes 2D. Nous pouvons également reconstruire la forme du visage avec une grande précision, le visage étant modélisé en termes de Composantes en Analyse Principale. Cette reconstruction est faite à partir de mises en correspondance de points d'intérêt et de silhouettes retrouvées automatiquement. Le suivi d'objets déformables grâce aux implicit meshes, également à partir de points d'intérêt et de silhouettes dans des séquences monoculaires, est démontré sur deux

exemples: La modélisation des déformations d'une feuille de papier représentée par un maillage triangulé classique, et le suivi des épaules d'une personne, dont les déformations sont exprimées en termes de DFFD.

To my Milena and our wonderful children Filip and Julijana.

Acknowledgment

First of all would like to thank my supervisor Prof. Pascal Fua for being an exceptional thesis advisor. Pascal was always there when I needed his help. His wide expertise and valuable advice helped me to overcome the challenges encountered during this research. It was always a pleasure to discuss and discover new solutions in a wonderful world of Computer Vision. This was an extraordinary experience for me and I will be always grateful for it.

I would like also to thank Prof. Daniel Thalmann in whose Virtual Reality Lab I started this work. I am thankful to the my committee members: Prof. Thomas Vetter, Prof. Radu Horaud, Prof. Daniel Thalmann and Prof. Wulfram Gerstner for accepting to judge this work. Their appreciation of my work confirmed that I was on a good path and motivated me to continue in the same direction.

Further, I must not forget my colleagues form the lab: Ralf Plänkers, Vincent Lepetit, Raquel Urtasun and Ali Shahrokni with whom I had a lots of fruitful discussion while looking for the solutions of problems arisen during the thesis. I also want to thank Pascal Lagger whose codes I used for generating some of the resulting images in this work. I have to thank Mathieu Salzmann with whom I partially worked on the last part of the thesis. A great gratitude I owe to Miodrag Dimitrijevic, my office mate, best friend and best man who probably knows all my work in details, since we spent hours discussing everything concerning this work. We also worked together on one part used in the thesis. I am grateful to the designers from VRLAB: Thierry Michellod, Mireille Clavien and Stephanie Noverraz as well, for creating some of the models I used in this work. Finally, I must not forget the Serbian community at the EPFL who was always a good company and made me feel as I am still in my home country.

I would like to thank my parents who gave me all the support and love during all my studies. I am grateful for always allowing and supporting me in everything I loved to do. Without them all this would be just a dream.

Finally, a big thank to my family, to my Milena and our children Filip and Julijana. I do not have the words to thank them enough for all the support and love they had for me during, sometimes, the endless working hours. Milena was not only supportive and full of understanding for my work, but also participating in many discussion concerning problems I was trying to solve. Her ideas were very useful, and the fact that I cloud discuss my work with her on the scientific level was exceptional and I appreciate it. Filip's smile and joy made me always forget work related problems in playtime with him. Julijana who was just born two moths before my thesis exam was an exceptional baby. She ease my final efforts by being calm, sweet and piecefull over the nights.

Contents

List of Figures

1 Introduction

In the last decades, Computer Graphics specialists have been looking for efficient methods to represent and manipulate 3–D objects. These models are usually created through a painstaking manual design process, using *Computer Added Design* (CAD) tools, or by reverse engineering of sculptured prototypes using modern scanning technology. Computer Vision practitioners have tried to offer an alternative to this expensive technology that relies on relatively cheap, passive sensors, such as ordinary cameras and automated algorithms. In the Computer Vision literature this is referred to as the *Shape from X* problem. Many reconstruction techniques, such as stereo, shape from silhouettes, space carving etc, have been developed. However, data obtained using those techniques are usually noisy and incomplete and the results cannot directly be used for further processing without first converting them to a representation that can be easily manipulated for further processing, such as animation. Model based approaches therefore appear to be more suitable. They usually involve fitting generic models of objects of interest to image data such as stereo, silhouettes and interest points.

Finding an efficient model surface representation that can take advantage of various image data observations is interesting and challenging problem. Deforming such surfaces in a generic fashion, which does not depend on the surface geometry and complexity, is also important problem. In this thesis we address a novel surface representation that we call *implicit mesh*, which can take advantage of various image data information, such as stereo, silhouettes and interest points, for efficient automated shape recovery. We also propose to use *Dirichlet Free Form Deformation (DFFD)* which provides parameterization of generic surfaces. It is suitable for both, automated fitting algorithms of Computer Vision and for free form shape modification required by Computer Graphics designers for surface editing and animation. The implicit mesh surface representation can be parameterized using DFFD, or it can be equally parameterized in terms of any other known surface parameterization. We demonstrated its applicability in attractive Vision problems such as 3D reconstruction and tracking of deformable objects.

1.1 Method Description and Applications

In the literature we can distinguish two main groups of surface representations that have been used to represents a priori given generic models of the objects whose shape we want to recover. These representations are known as explicit and implicit surfaces. Explicit surface representations are intuitive and easy to manipulate, and they are widely accepted among graphics designers. However, they are not necessarily ideal for fitting surfaces to po-

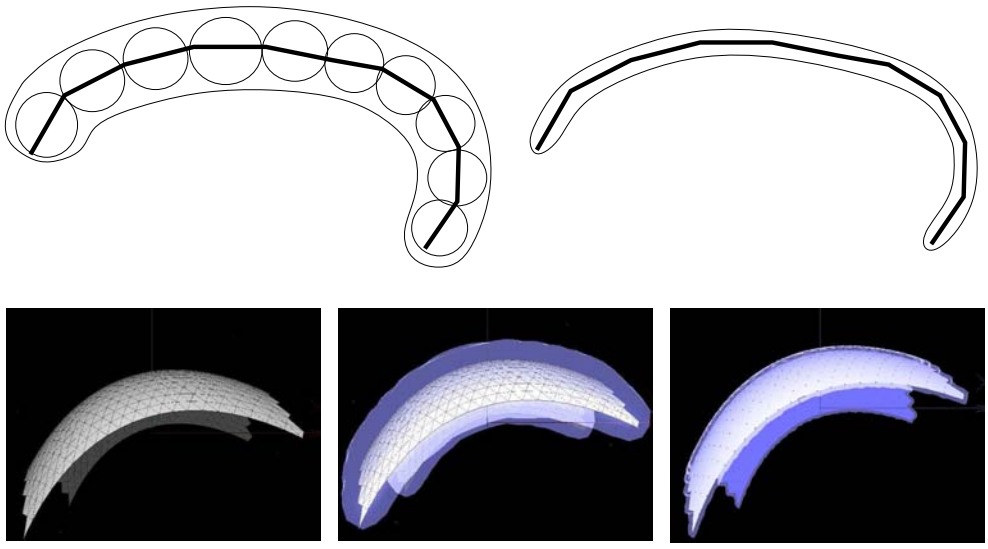


Figure 1.1: Turning the explicit surface into implicit mesh. Top row: Simplified 2D case of creating the implicit mesh by attaching spherical metaball to each line segment on the left and by attaching more sophisticated triangular metaball on the right. Bottom row: Converting real 3D explicit mesh into spherical, shown in the middle, and triangular implicit mesh, shown on the right. Look at the Chapter 4 for detailed discussion on implicit meshes.

tentially noisy and incomplete data such as 3–D points produced by stereo systems or 2–D points from image contours, because fitting them involves minimizing a non-differentiable distance function. Implicit surfaces are well-suited for simulating physically based processes [115, 117, 83, 88] and for modeling smooth objects [2, 118, 40]. Because the algebraic distance to an implicit surface is differentiable, they do not suffer from the drawbacks discussed above when it comes to fitting them to 2 or 3–D data [110, 90, 31].

In short, explicit surface representations are well suited for graphics purposes, but less so for fitting and automated modeling. The reverse can be said for implicit surface representations. In this thesis, we propose to combine the strengths of both approaches to avoid their drawbacks, and in this way build robust surface representation suitable for automated shape recovery from video sequences. This is done by:

1. transforming explicit surfaces into implicit ones, whose shape closely approximates that of the original triangulations

2. deforming the implicit and the explicit surfaces in tandem for fitting and rendering purposes

As shown in Fig. 1.1, to create these *implicit meshes*, we attach triangular or spherical metaballs to each facet of the explicit mesh. The implicit mesh can be simply created

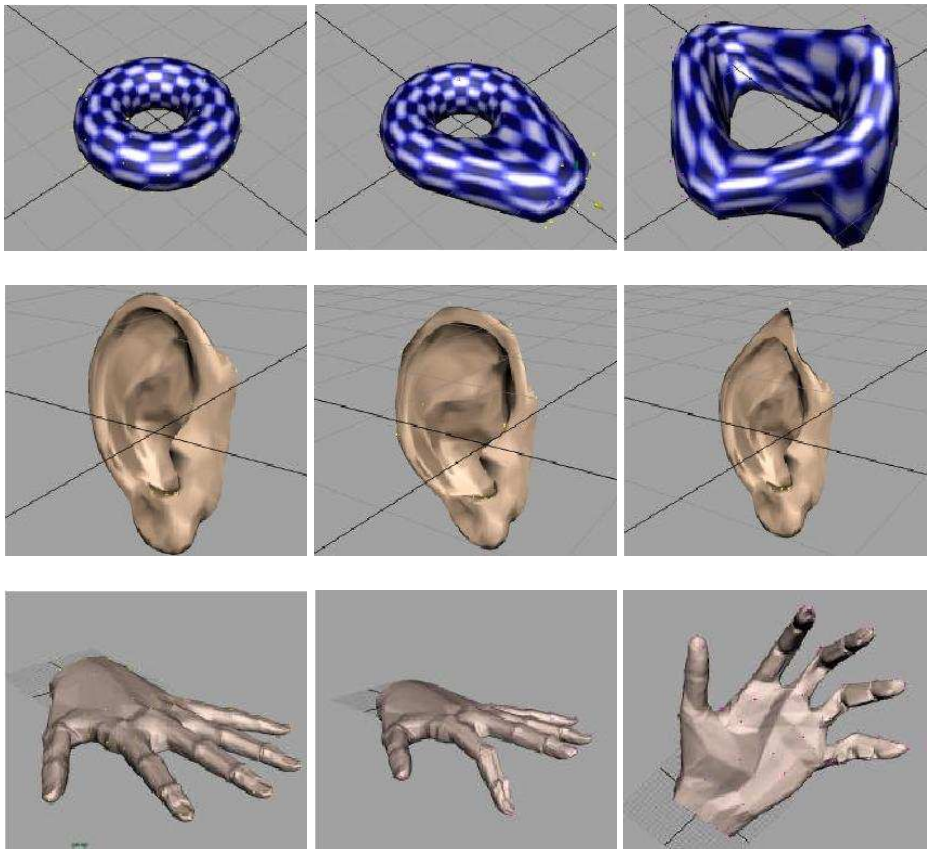


Figure 1.2: Examples of the Dirichlet Free Form Deformation (DFFD) applied to the objects of different geometry and complexity indicating its generality. In the first column initial shape of the objects of different geometry are shown, followed by their deformed versions obtained by appling DFFD to the their initial forms. The deformation results are generated by Maya plug-in which we implemented for DFFD.

from any polygonal or parametric surface by wrapping a thin shell around it. This shell actually is an iso-surface of the potential field function, that represents implicit surface. It is generated from the underlying explicit surface skeleton. The surface produced in this way retains the shape and parameterization of the original explicit surface. The shape of this surface depends *only* on the underlying explicit surface geometry. As a result, when a explicit surface deforms, so does the corresponding implicit surface.

To simultaneously control the overall shape of the explicit and implicit meshes, we use Dirichlet Free Form Deformations [84]. This kind of deformation appears to be generic, and can be applied to the object of any geometry and topology as shown in Fig. 1.2. In this way we have a tool for generic free form object modeling and deformation. The DFFD

approach involves an arbitrary object model whose shape can be changed by moving a number of control points. Here, DFFD control points can be placed at arbitrary locations—that is, on the object, inside of it or outside—rather than on a regular lattice, as it is the case in most Free Form Deformation (FFD) [100] approaches. In particular, some of the control points can be important feature points that must be controlled in a specific way or in general can be taken to represent decimated version of the surface we want to deform. It is generality achieved by using generalized natural neighbor coordinates, also known as Sibson coordinates[104] and using a generalized interpolant [42]. As a consequence surface model vertices appear to be a weighted liner combination of local subset of the control points. Expressing complex model surfaces in terms of much smaller number of control parameters is a desirable property for optimization algorithms.

Because the shape of the implicit mesh strictly is a function of that of the explicit one, we could use any other parameterization applied to the explicit mesh. To demonstrate this we parametrized them both, directly in terms of the 3–D coordinates of their vertices and in terms of PCA weights [7] that control the shape of the model. This means that the implicit meshes are independent of the way the initial model is parameterized.

Our contribution is therefore twofold. First, we propose an approach to surface reconstruction that lets us take an explicit surface model of arbitrary complexity and regularity, turn it into an implicit mesh, and take advantage of the attractive properties of implicit surfaces for fitting purposes. Because the implicit surface closely approximates the explicit one and they deform together, the reshaped explicit mesh is also available for rendering and animation. This lets us handle arbitrary triangulations that were not necessarily designed with fitting in mind. Second, we propose a generic model deformation approach based on DFFD method. It allows to take any surface model represented as an explicit mesh, create a control mesh as, either decimated version of the surface model or as a collection of characteristic points of the surface model, and deform it by moving the control points around. In terms of fitting the automated optimization procedure will provide the optimal control points positions, such that deformed model confirms to the data. Such model can be further used for additional editing and animation using DFFD.

We demonstrated power of implicit mesh representation in the task of 3–D shape recovery from images of the upper body, parametrized in terms of DFFD. The images are coming from uncalibrated video sequence, which is registered thanks to the explicit part of our model. Further, triangular implicit mesh is used for fitting it to stereo and silhouettes simultaneously. In the top row of Fig. 1.3 we showed two frames of the original sequence with overlaid silhouettes on the shoulders, together with the textured reconstruction results shown besides the originals. Note that the reprojection of the reconstructed model correctly aligns with the silhouettes indicating precision of the shape recovery. We also used high resolution PCA face models to simultaneously recover the shape and camera motion. In this case we used corresponding interest points between consecutive frames and automatically detected silhouettes, as explained bellow, to fit our model to them as it is shown in the second row of Fig. 1.3. Two original images and the reconstructed textured models obtained using both interest points and silhouettes are shown together with the automatically

detected silhouettes.

In this work we were especially interested in efficient handling of occluding contours, that appear to be a key clue to recovering the shape of smooth and potentially deformable surfaces in monocular sequences. However, because extracting them reliably against potentially cluttered or changing backgrounds such as those of the third and bottom row of Fig. 1.3, is difficult, most of the published work involves engineering the environment to make this task easier. In this work, we show that representing generic 3-D surfaces as implicit meshes allows us to automatically detect silhouettes in the images and to take advantage of occluding contour constraints. Furthermore, it also lets us effectively combine silhouette information with that provided by interest points that can be tracked from image to image. This is important because this may mean the difference between the ability or the inability to exploit silhouettes in uncontrolled real-world situations where occlusions and difficult backgrounds often degrade the output of even the best edge detection algorithms.

More specifically, the *implicit meshes* [61], allows us to robustly detect the occluding contours on the 3-D surface as the solution of an ordinary differential equation [95]. Their projections can then be used to search for the true image boundaries and deform the 3–D model so that it projects correctly. This well-formalized approach yields a robust implementation that we demonstrate for monocular tracking of deformable 3–D objects in a completely automated fashion: We start with a generic 3-D model of the target object, find its occluding contours, and use them to search for the corresponding contours in the images. We then use the detected 2-D contours and the constraints they impose, along with some feature information when available, to deform the model. On this procedure we based our tracking algorithm that successively fits the deformable objects following nonrigid forms of the objects.

This approach is effective independently of the specific way the deformations are parametrized. As shown in Fig.1.3, we validated the tracker in two very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations [84]. The results of paper tracking in case of changing background and partial occlusions is shown in the bottom row of Fig. 1.3. The examples of deformable shoulders tracking against the cluttered background are shown in the third row of Fig. 1.3.

We want to highlight that produced surfaces can be easily manipulated and animated by the Computer Graphics practitioners. Results of deformable surface tracking appear to be good clue for automatizing animation procedure, which might require days of manual reshaping. Also, tracking of the structures that have certain physical properties, like a piece of paper deformation, can help extracting important material properties and forces applied to produce those deformations. This all can be suitable in entertainment industry, robotics, structure mechanics of big deformations, or for augmented and virtual reality applications.

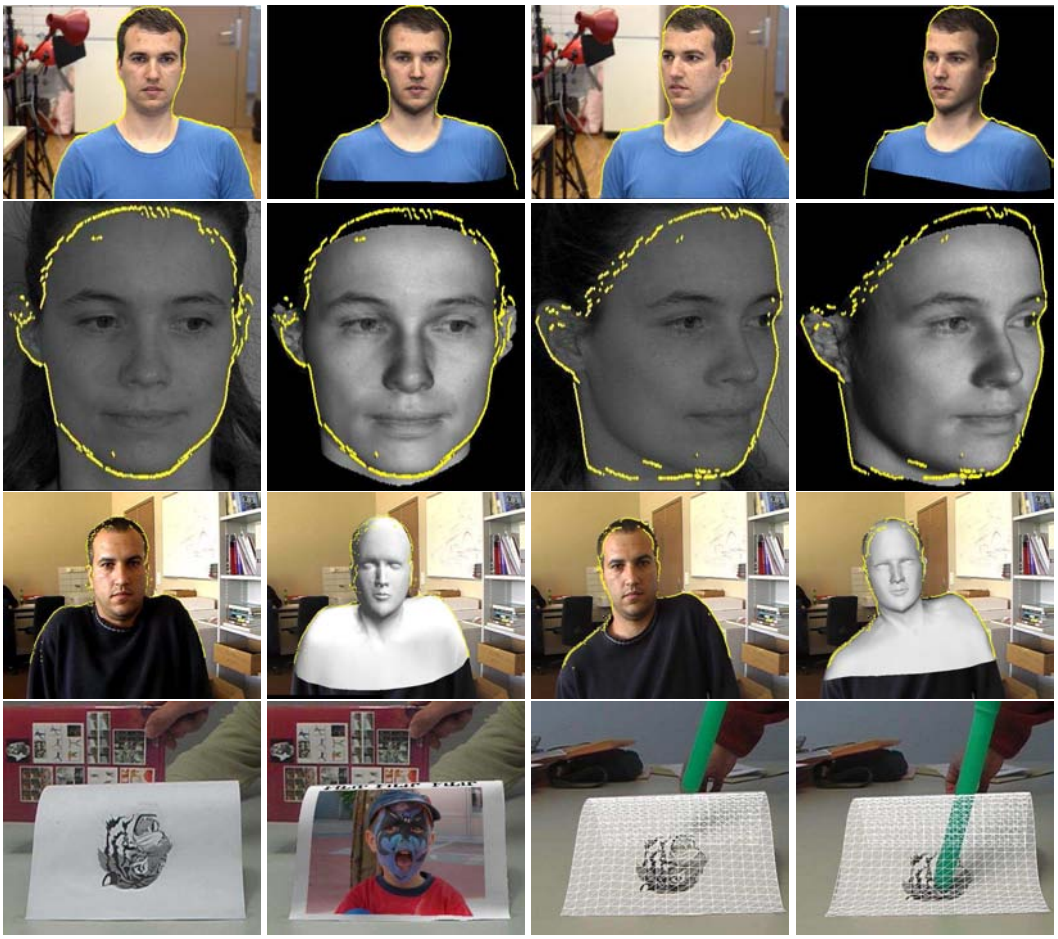


Figure 1.3: Reconstruction and tracking results obtained using implicit mesh formalism. Top row: 3D reconstruction of the human upper body, parametrized in terms of DFFD transformation, from uncalibrated video sequence using stereo and silhouette information. Original images and textured reconstructed models are shown together with overlaid silhouettes. Model alignment with the silhouettes and correct texture map indicate quality of the results. Second row: Face reconstruction using PCA face model. First and third images are two input images with overlaid silhouettes automatically detected using PCA face model turned into implicit mesh. The images next to them depict textured reconstruction result. Note that the camera motion is recovered together with the shape. Third row: Shoulders and head tracking in a cluttered scene using implicit meshes highlights its ability to handle occluding contours in a robust way. Bottom row: Tracking of deformable piece of paper in presence of the changing background and partial occlusion. Silhouette information is again handled using implicit meshes.

1.2 Goals and Contributions

This work introduces novel surface representation called *implicit mesh* that combines good properties of both explicit and implicit surface representations for automated shape recovery from images. We focus on an approach to model based surface reconstruction, where generic surface models of arbitrary geometry and complexity, can be taken and converted into implicit meshes. The constructed implicit mesh model allows efficient fitting to the various data sources coming from images, such as stereo, silhouettes and interest points. We demonstrate all this in the challenging context of monocular video sequences where the motion of the camera usually is not necessarily known a priori. In this case, existence of the universal and robust generic models, which can take advantage of as many as possible image clues simultaneously and in completely automated fashion, is very important. We also proposed to use a generic approach to automated surface deformation based on DFFD technique.

Many model based surface reconstruction and tracking approaches are based on purpose built models. For example, approaches to face and body reconstruction require specially designed and parametrized face and body models. By contrast, our implicit meshes do not depend on the way the surface is parametrized. Furthermore, the models we use do not have to be specially designed either with fitting or with some other specific application in mind. In practice, we can take any polygonal surface model, even from the web, turn it into our implicit mesh representation and use it for tracking and shape recovery. This representation brought us possibility for the automated handling of occluding contours in the hard cases of cluttered background, partial occlusions and background changes.

To summarize, implicit meshes as a novel surface representation approach have following advantages:

- **Generic surface representation that unifies good properties of both, explicit and implicit surfaces representations:** Easy manipulation, visualization and possibility of modeling objects with complex geometry are inherited from the explicit surfaces, and compact representation in terms of just one mathematical function, differentiability and ability of fitting to the noisy data are inherited from the implicit surfaces.

- **Generic approach to the surface deformation:** Dirichlet Free Form Deformation (DFFD) technique is used as a generic approach to object deformation that can be applied to the objects of arbitrary complex geometry and topology.

- **Effective use of various data sources, such as stereo, silhouettes and interest points, separately or simultaneously:** The implicit meshes induce the use of differentiable distance function, and its dual representation allows alternative use of either explicit or implicit part of the surface representation for fitting.

- **Effective silhouette detection and their handling in the complex environments:** Presence of the implicit surface allows automatic computation of the occluding contour of the surface model, that is further used for automatic detection of the object's

outlines in the images. Presence of the model is a strong clue that produces robust silhouette handling in complex cluttered scenes, changing backgrounds and partial occlusions, which definitely makes a difference compared to known methods.

- **Independence from the surface parameterization:** No matter how the initial explicit surface is parametrized, the implicit mesh inherits this parameterization, and allows implicit surface deformation in tandem with the explicit skeleton.

Finally, we tested implicit meshes for surface reconstruction from uncalibrated video sequence of the human upper body including – head, neck and shoulders – parametrized in terms of DFFD transformation. We also reconstructed the human ear, from the pair of images under the structured light. The PCA face model was used for object shape reconstruction and camera motion recovery from images. Tracking examples include upper body motion tracking and tracking of the deformable piece of paper in monocular scenes.

Parts of this work have been published in various international conferences including: [59, 60, 62, 61, 34, 63]

1.3 Thesis outline

In the reminder of the thesis, we first present related approaches to surface representation in order to compare them with our surface representation, which is given in Chapter 2. In Chapter 3, we introduce the Dirichlet Free Form Deformation (DFFD) as a generic approach to surface deformations, and give the examples of their efficient use for object reconstruction from images, shape modification and animation. In Chapter 4, we discuss how the implicit meshes are constructed and what are their properties. Optimization framework is presented in Chapter 5, comparing our method to fitting with explicit meshes. Here we present fitting to stereo and silhouettes appearing in various application where we tested our method. Automatic silhouette handling using implicit meshes and fitting using interest points is discussed in Chapter 6. The results of our reconstruction and tracking experiments are presented in Chapter 7, followed by the conclusion.

2 State of the Art

This thesis addresses the problem of building a robust surface model representation that can be used for efficient 3–D reconstruction from uncalibrated video sequences and tracking of completely nonrigid 3–D object deformation in monocular scenes. In the following sections we review the various surface representations. We also discuss state-of- the-art in using those representations for automated 3–D reconstruction and tracking of deformable objects.

We will begin with the overview of surface model representations used for 3–D reconstruction. Then, we will address the problem of tracking deformable object.

2.1 Surface Model Representations for 3D Reconstruction

In the last decades, Computer Science specialists from various domains, have shown broad interest in representing three dimensional objects using computers. More specifically, Computer Graphics specialists have been looking for efficient methods to represent and manipulate 3–D objects. Advances in computer speed, memory capacity and hardware graphics acceleration have made model representation and manipulation feasible. These models are usually created through painstaking manual design process, using *Computer Added Design* (CAD) tools, or by reverse engineering of sculptured prototypes using modern scanning technology.

Even though Computer Vision practitioners sometimes dealt with active sensors, such as laser scanners, they tend to focus on offering an alternative to this expensive technology that relies on relatively cheap, passive sensors, such as ordinary cameras. In the Computer Vision literature this is referred as *Shape from X* problem. A popular technique of reconstruction from two or multiple camera views is stereo, where actual depth information is recovered [43, 54]. Another approach is to use silhouette contours to recover shape from multiple views [19]. Adding a photo consistency constraint to the shape from silhouettes results in a the technique called *space carving* [72], where photo-consistency means that the valid point in the scene surface appears with the *same* color over all images that are visible, under the assumption that the surface is Lambertian. Data obtained using those techniques are usually noisy and incomplete. In this thesis, in part, we will concentrate simultaneously on shape recovery from stereo and silhouettes given a priory knowledge of the object we want to reconstruct.

There are two main directions in 3–D reconstructions. One is direct reconstruction, already mentioned under the name Shape from X, of the scene without using predefined models where the structure is retrieved directly from the data (either images or laser scanned

3–D points). The other is model based reconstruction where we use the a priori knowledge of the kind of objects we want to reconstruct and then fit such models to the data obtained either from images or from Shape from X techniques. In Computer Graphics, where researches deal more with the laser scanned data, various techniques of reconstructions from unorganized data clouds are established. In Computer Vision model based reconstruction is more spread, because of noisy and incomplete data produced by vision techniques.

For all surface models that we investigate, it is possible to define following criteria relevant for shape recovery:

- *Local Control* or *Deformability* – represents ability of surface model to be easily and intuitively deformed either by the graphics designers or some algorithms that control the shape of the object
- *Easy Visualization* – concerns simple and well established rendering algorithms
- *Differentiability* – relates to the surface differential properties, which is important when automated optimization procedures are applied for reconstruction and tracking
- *Fitting noisy data* – highlights the ability of certain representation to fit extremely noisy and incomplete data such as stereo obtained by vision techniques
- *Modeling object of complex geometry* – underlines model's ability to represent objects of any free form shape

In both Computer Vision and Computer Graphics various surface model representations are used. We can divide them in two main groups: explicit surfaces, implicit surfaces. In the following sections we will discuss those surface representations giving their brief description and overview their use in 3–D model reconstruction and tracking.

2.1.1 Explicit Surface Models

Explicit surface models are models that can be explicitly defined by certain complete mathematical forms. In this section we will describe different types of explicit surface models and overview their use in 3–D reconstruction and modeling.

We will start with polygonal meshes that are the most frequent in both vision and graphics, then continue with parametric surfaces. Further, we will discuss superquadrics as very popular representation in many vision applications that are actually a special class of parametric surfaces. Another parametric representation is based on B-spline basis functions. As last in a row of parametric surfaces are discussed generalized cylinders. Finally, in recent years a very popular way of representing surfaces in Computer Graphics community are the subdivision surfaces.

2.1.1.1 Polygonal meshes

Definition The most popular way of representing 3–D surfaces is the polygonal mesh. A polygonal mesh $\mathcal{M}$ can be defined a pair of ordered lists:

$$\begin{aligned} \mathcal{M} &= \langle \mathcal{V}, \mathcal{F} \rangle \\ V &= \{\nu_{\mathbf{1}}, \nu_{\mathbf{2}}, \ldots, \nu_{\mathbf{N}_\nu}\} \\ \mathcal{F} &= \left\{\phi_{\mathbf{1}}, \phi_{\mathbf{2}}, \ldots, \phi_{\mathbf{N_p}}\right\} \end{aligned} \tag{2.1}$$

where V is a list of N_ν three-dimensional vertices $\nu_i = [x_i, y_i, z_i]^T$ and $\mathcal{F}$ is a list of polygons or facets each specified as a list of vertex indices $\phi_i = \left\{\nu_{\mathbf{i},1}, \ldots, \nu_{\mathbf{i},\mathbf{n}_{\nu_\mathbf{i}}}\right\}$. If $n_{\nu_i} = 3$ the polygonal mesh is composed of triangles and we call it triangular mesh or *triangulation.* This kind of polygonal representations is the most common in the literature. Thanks to the convexity of the triangle the simpler rendering algorithms and compact memory representations are established. Further in this work we will refer to triangular polygonal mesh as *mesh.*

Properties Given the definition of the mesh we can discuss properties of such representation that are already mentioned above. Polygonal meshes can be easily deformed either by directly pulling their vertices or by using some algorithms that displace vertices by deforming the volume in which the mesh is enclosed. Enclosing polygonal mesh usually involves putting control grid, composed of bunch of control points, around the mesh and moving those control points freely in space. The volume they enclose is warped, which is applied to the mesh inside of it. This will be discussed more in the following section where the model parameterization is addressed. Various rendering algorithms are implemented for fast polygonal mesh visualization. In recent years they are even implemented in hardware to accelerate rendering of huge meshes. Meshes are widely used for fitting noisy data. However, using them involves attaching observations to the closest facets that involves search, which is slow in case of high resolution meshes. Also they suffer from the non-differentiable distance function when it comes to the change of the attachments among the facets during the optimization. From our practice it turned out that fitting them to silhouette data is where they usually fail. Finally, meshes can be used to model objects of any geometry.

Literature overview Polygonal surface models are used in many contexts like structure from motion, head modeling, body modeling, modeling of architectural objects and modeling of physically based dynamic objects. Bundle-adjustment is a popular structure from motion technique to refine, both 3D structure and camera motion, from the image sequence. Objects/scenes can be represented either as point clouds or geometric models. Even though a point cloud is a more general representation of the modeled scenes it has to address, yet generally unsolved, vision problem of establishing reliable feature correspondences across

the images. However, using the geometric models allows to additionally constrained the problem and make it a better posed.

One of the pioneer works that uses complex polygonal meshes for fitting and tracking is work of David Lowe [80]. The complex hand drilling machine is represented as set 3–D polygonal meshes each of which has six degrees of freedom. The shape of the tool is adapted to project correctly in the image. Further the moving parts are tracked by minimizing distance of the model's occluding contour toward the image edges using Levenberg-Marquardt [93] optimization algorithm.

Fua [47] used generic animation mask of a human head, represented as simple mesh, to recover camera motion and rough shape of the face through model-driven bundle adjustment procedure. The underlying model is also used for regularized optimization procedure so that resulting mesh does not go too far from the generic model. The obtained resulting model is additionally fitted to the stereo data cloud, computed according to the previously registered images, in order to improve the quality of the result. Complexity of the optimization procedure depends on the models complexity, i.e on the model's mesh resolution. This problem is addressed in work of Ilic and Fua [59] where Dirichlet Free Form Deformation (DFFD) is used mainly to reduce model's complexity. In [49] besides stereo data cloud the silhouette information is used to correct the fitting result at the occluding contours. From this work it turned out that searching for the silhouette facets is hard, depends on the model resolution and results in an imprecise results.

Shan at al [102] presented a method for model-based bundle adjustment for face modeling with the polygonal mesh as a generic face model. Actually, face mode is represented as a liner combination of the neutral face and number of face metrics representing vectors that linearly deform a face in a certain way, such as to make head wider, nose bigger etc. This diminishes number of model parameters to optimize. Here the model is not used as a regularizer, but it is used directly as a search space that results in an elegant mathematical formulation. It involves transfer function that actually back projects feature point from one image frame over the 3D polygonal model to the subsequent image frame. However, the algorithm requires setting 3–D feature points whose projections are known, which, in practice, results in a relatively complex processing chain.

Kang et al [69] proposed appearance based structure from motion using a linear combination of 3–D meshes of scanned human faces as their model space. The model space is again used as regularization constraint to ensure that the resulting face is still close to the human face. This model involves huge number of parameters to optimize and lots of constraints leading to the ill-posed problem, thus more difficult to achieve convergence. One of the ways to overcome this problem is proposed by Blanz and Vetter [7], where the statistical model based on Principal Component Analysis was used to reduce dimensionality. It includes shape and texture components that have been learned from the database of scanned human faces. However, demonstrating excellent results, this approach might have problems when dealing with images containing cast shadows and strong specularities. Another effective solution is proposed by Dimitrijevic et al [34], where texture component of the model is replaced by a set of 2–D correspondences in all pairs of consecutive images. This

is similar to [102], but with the increased automation by eliminating need for 3–D feature points whose projections are known. Also, the high resolution model leads to the visually more pleasing results.

Another work on structure from motion problem concerns work of Debevec et al [26] that recover architectural models using polyhedral geometric primitives. Those primitives represent basic architectural construction blocks, like cubes, prisms, pyramids etc. They are parametrized in terms of their dimensions including length of their edges. Once the model is reconstructed and images are registered, a model based stereo algorithm is run to refine fine architectural details like windows, friezes and cornices to actually conform to its actual appearance in a set of photographs.

Besides widely spread face modeling application, systems proposed by Hilton et al [56] and Lee et al [74] allow for automatic or semi-automatic body modeling from three or four orthogonal photographs. Both approaches involve complete polygonal body models and have as a goal achievement of low-cost modeling of individual people for realistic computer generated imagery in virtual worlds.

Work of De Carlo and Metaxas [27] on face modeling and tracking is based on dynamic face model represented as a mesh. The optical flow is used to constrain motion of the deformable model. Face model is specially designed to respond to the certain set of physically based deformations. Works of Cohen [20] and Terzopoulos [116] modeled dynamic objects as the polygonal meshes and used them for 3–D reconstruction. Models are deforming according to the physical properties of elastic objects.

As it can be seen from the literature overview that polygonal meshes appear as an indispensable models, no matter how they are parametrized and in which context are used.

2.1.1.2 Parametric Surfaces

Parametric surfaces are another type of 3–D surfaces that represent generic surface shapes as a 2–D manifolds embedded in 3–D given the set of variable parameters. A generic mathematical formulation of 3–D parametric surface is:

$$S(u,v) = (X(u,v), Y(u,v), Z(u,v)) \tag{2.2}$$

where *u* and *v* are varying parameters that can be limited to certain interval, e.g a unit square $[0,1] \times [0,1]$. In Computer Vision, most frequently used parametric surfaces, are: superquadrics, Bézier and B-spline surfaces and generalized cylinders. These surfaces were used in various contexts such as model reconstruction, object recognition and classification, segmentation, 3-D modeling and image compression. Equally in Computer Graphics these representations became parts of standard modeling softwares. Now, we will discuss each of these representations separately by giving their definition, describing their properties in favor of automated reconstruction and tracking and finally give the literature overview of their use by the researchers worldwide.

The Superquadrics

Definition Superquadrics appear as the generalization of the quadric surfaces first introduced in Computer Graphics by Alan Barr [5]. Quadrics, like ellipsoids, hyperboloid, paraboloid etc., are also used in Computer Vision, but because of their limited modeling capacity they were soon replaced by more general superquadrics. The actual extension from quadrics to superquadrics is made by introducing "bulge factors" ϵ_1 and ϵ_2 to which various terms are raised. The fact that superquadrics appear in parametric and implicit mathematical form made them suitable for both, efficient and easy rendering for graphics purposes, and automated model recovery from images, for vision purposes. The interested reader can find the excellent book of Ales Jaklic et al [65], on superquadrics and their use in computer vision. The general parametric and implicit mathematical formulation for the superellipsoid, centered at the origin and with the axes aligned with the axis of coordinate system, are given respectively by:

$$Q(u,v) = \begin{bmatrix} a_1 \cos^{\epsilon_1} u \cos^{\epsilon_2} v \\ a_2 \cos^{\epsilon_1} u \sin^{\epsilon_2} v \\ a_3 \sin^{\epsilon_1} u \end{bmatrix}, \quad \begin{matrix} -\pi/2 \leq u \leq \pi/2 \\ -\pi \leq v \leq \pi \end{matrix} \tag{2.3}$$

$$Q(x,y,z) = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\epsilon_1}} - 1 \tag{2.4}$$

Parameters a_1, a_2 and a_3 are scaling factors along the three coordinate axes. ϵ_1 and ϵ_2 are exponents of the two original superellipses whose spherical product actually led to Eq. 2.3. ϵ_2 determines the shape of the superellipsoid cross section parallel to the xy-plane, while $\epsilon_1$determines the shape of the superelipsoid's cross section in a plane perpendicular to the xy-plane and containing z axis. Note that when the exponential coefficients are $\epsilon_1 = \epsilon_2 = 2$ we obtain a special case where the superquadric is transformed into quadric. In this case we will obtain an ellipsoid.

Properties As already mentioned superquadrics are very easy to render. Thanks to their parametric form it is straightforward to sample them and convert them into the polygonal meshes that are easy to render. On the other hand, because of their implicit formulation, superquadrics are suitable for model recovery from noisy range data, since the distance function to minimize can be expressed as simple differentiable algebraic distance function of 3–D observation points to the surface. Beside simple algebraic distance it is possible, depending on the application, to use different distance metric as studied by White and Ferrie in [122]. Also, superquadrics can be fitted to 2–D data such as silhouette edges. They are good at capturing global coarse shape of the 3–D objects. Introducing global deformations, such as tapering, twisting and bending it is possible to increase the expressive power of superquadrics, but they are still limited to the global coarse shapes as opposed to the local details. This can be remedied by adding local degrees of freedom. However, in that case we end up with a large number of degrees of freedom that simply over-parameterize the

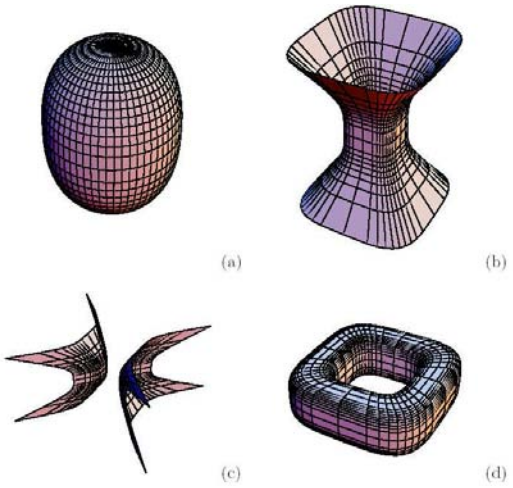


Figure 2.1: Superquadric of different shapes: (a) superellipsoid (b) superhyperboloid of one, and (c) two sheets and (d) supertoroid. The figure is courtesy of [65].

model. Recovering simple structures becomes hard in this case. Finally, superquadrics can be efficiently used to model objects of limited complexity. In recent years, objects like articulated structures of human body and human hand have been successfully represented using those primitives.

Literature Overview After introducing superquadrics in Computer Graphics by A. Barr [5], Alex Pentland brought them to the Computer Vision community [89]. He proposed, similar to A. Barr, to use superquadrics models in combination with global deformations like tapering, bending and twisting, for describing a scene structure which can be recovered from images. Further, researchers like Gupta and Bajcsy [51], Solina and Bajcsy [106], Raja and Jain [94] addressed problem of shape recovery from pre-segmented range data. On the other hand Pentland [86], Leonardis et al. [76] tried to solve 3–D segmentation problem together with shape recovery of simple objects in the scene. Other researcher, like Sclaroff and Pentland [99], Metaxas and Terzopoulos [83], and DeCarlo and Metaxas [28], used superquadrics for dynamic, physically based simulations and shape recovery. Recently, superquadrics and quadrics became indispensable part of almost every articulated model, such as human body and hand. We will mention works of D. Gavrila et al. [50] for articulated human motion recovery in multiple camera environment, then tracking of human body in monocular case by Sminchisescu and B. Triggs [105]. CAD system for garment industry (Jojic and Huang 2000 [67]) recovered shape of human body parts, represented by superquadrics enhanced by local deformation, from 2–D images and stereo, added by structure light. Also, hand model made of superquadrics for tracking was proposed by Stenger et al. [107].

Smooth B-spline Surfaces

Definition Spline curve $P(t)$ is piecewise polynomial defined on a knot vector $T = [t_0, t_1, t_2, \ldots]$. Elements of the knot vector are values of the curve parameter t where the change of the polynomials occur, i.e. where the polynomials join together making a piecewise polynomial. A *B-spline* curve $P(t)$ of order m, with the knot vector T and $L+1$ control points P_k, $k = 0, \ldots, L$ is given by:

$$P(t) = \sum_{k=0}^{L} P_k N_{k,m}(t) \tag{2.5}$$

where P_k are control points and $N_{k,m}(t)$ are *B-spline* basis functions of degree $m-1$, that are given by the recursive definition:

$$\begin{aligned} N_{k,1}(t) &= \begin{cases} 1 & , if\ t_k < t \leq t_{k+1} \\ 0 & , otherwise \end{cases} \\ N_{k,m}(t) &= \left(\frac{t - t_k}{t_{k+m-1} - t_k}\right) N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}}\right) N_{k+1,m-1}(t) \end{aligned} \tag{2.6}$$

To illustrate this we show in Fig. 2.2(b) an example of B-spline curve based on eight control points and cubic $(m = 4)$ B-spline blending functions shown in Fig. 2.2(a). The knot vector $T = (t_0, t_1, ..., t_{L+m})$ is created with multiple knots at the beginning$(t_0 = t_1 = ... = t_{m-1} = 0)$, and the end$(t_{L+1} = t_{L+2} = .. = t_{L+m} = 0)$ so that first and last control points are interpolated, i.e. curve starts from the first control point and ends at the last control point. Knots $t_m, \ldots, t_L$ increase in increments of 1, from 1 to the value of $L - m + 1$. Such knot vector is called standard knot vector.

From above given definition of the B-spline curve we can easily define B-spline surface as a tensor product of two B-spline curves, given like:

$$P(u, v) = \sum_{i=0}^{M} \sum_{k=0}^{L} P_{i,k} N_{i,m}(u) N_{k,n}(v) \tag{2.7}$$

where $P_{i,k}$ make a control polygon, while $N_{i,m}(u)$ and $N_{k,n}(v)$ are B-spline basis function (possibly of different orders m and n). An example of one such B-spline surface is shown in Fig. 2.2(c). To model complex objects using one B-spline patch will lead to the creation of extremely complex knot vector that became impractical to handle. For that reason complex geometric objects are created by stitching B-spline patches together, similar to one depicted in Fig. 2.2(d), taking care on surface continuity at the stitches.

A very popular generalization of B-spline surfaces are NURBS (non-uniform B-spline surfaces) whose principal advantage over the B-spline surfaces is that they can model exactly quadric surfaces and that they are invariant to the perspective projection. A NURBS surface can be in general defined as:

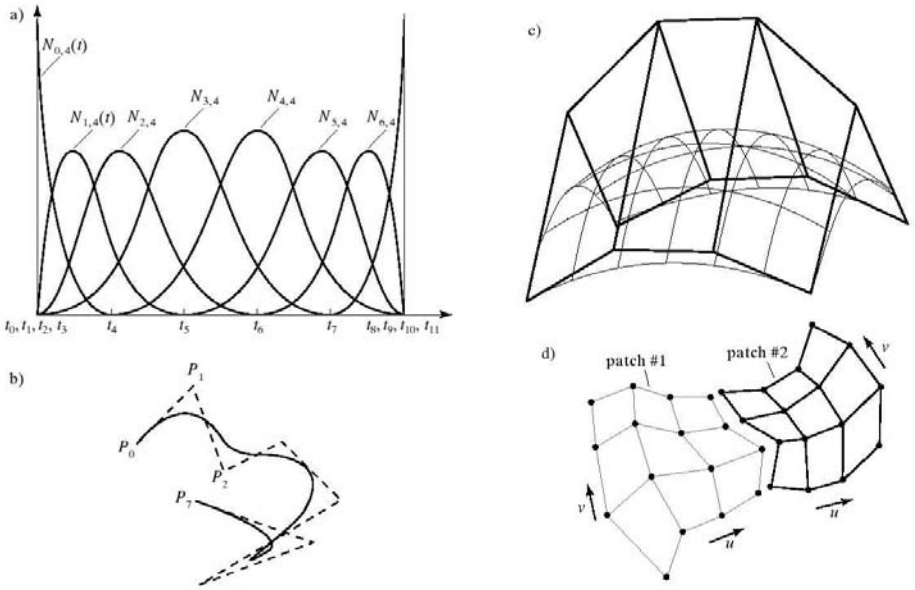


Figure 2.2: B-spline curve: (a) Cubic B-spline basis functions. (b) B-spline curve with eight control points where the first and the last are interpolated. (c) B-spline surface (d) Stitching continuously two B-spline patches. The figure is taken from [55].

$$P(u,v) = \frac{\sum_{i=0}^{M}\sum_{k=0}^{L} w_{i,k} P_{i,k} N_{i,m}(u) N_{k,n}(v)}{\sum_{i=0}^{M}\sum_{k=0}^{L} w_{i,k} N_{i,m}(u) N_{k,n}(v)}$$

where $w_{i,k}$are user defined weights, while the other elements are the same as for B-spline surfaces. Actually, setting all weights to one, the B-spline surface definition is obtained.

Properties As all parametric surfaces, smooth B-spline surfaces are easy to visualize. Smooth B-spline or NURBS surfaces demonstrate great local control of the surface shape. This is because of limited support of B-spline basis functions on the knot vector. Generally speaking, the overall flexibility of the surface depends on the definition of the knot vector. In theory, a very complex knot vector can be created so that B-spline surface can model any kind of shape. However, in practice it is quite difficult to design such knot vector and, for modeling objects of complex geometry, the B-spline surface patches network have to be created. Each surface patch is a C^{m-1} differentiable, where m is order of B-spline piecewise basis function. In case of joining surface patches, the continuity at the joints has to be preserved. Surface continuity can be simply achieved by duplicating control points at the joints as it is shown in Fig. 2.2(d). However, such surfaces have only geometric G^1 continuity, that is sufficient for most design applications. Achieving parametric or C^n continuity is not trivial and requires more sophisticated computation. Parametric continuity

is important in automated reconstruction tasks where optimization algorithms require those derivatives to be provided. Also, for modeling objects of arbitrary complex geometry, one will need a model with lots of surface patches. This means many control points, i.e. many parameters to optimize. If fitted to noisy data, as here we are interested in, to preserve continuity among patches sophisticated constrained optimization schemas are required.

Let us look in more details the problem of fitting B-spline based surfaces. We want to retrieve optimal position of the control points which compose a state vector $\mathbf{S} = [\mathbf{P}_{i,k}]^T$, $i = 0, M$, $j = 0, L$, by minimizing distance between B-spline surface and observation points $\mathbf{obs}_i$. The objective function has the following form :

$$\min_{\mathbf{S}} \sum_{i=1}^{n} \|P(u_i, v_i, \mathbf{S}) - \mathbf{obs}_i\|^2 \tag{2.8}$$

Note that for each observation $\mathbf{obs}_i$ point we have to know its position in parameter space (u_i, v_i). This is not a trivial problem and it is called parameterization of the sample (observation) points. If the samples can be organized to make a grid then exist technique for uniform parameterization of the sample points [81]. In practice sample points are usually noisy and unorganized. Ma and Kruth [81] proposed a method that projects sample points on the rough approximation of the surface to be fitted. The values of the base surfaces' local parameters of the projected samples are then used as the parameter values (u_i, v_i) in the fitting process. This can be iterated using the fitted surface as the base surface in the next iteration. Nice book on fitting spline curves and surfaces is written by Paul Dierckx [33].

Literature Overview There has been a long history of using B-spline parametric surfaces for automatic object reconstruction from unorganized 3–D data. Kind of data used for shape recovery range from laser scan and range images to occluding contours. Fitting those kind of surface representations simultaneously to the stereo and occluding contours we did not meet in the literature.

We will start by citing work of Zhao et al. [130] where they used bicubic regularized B-spline patches for global reconstruction of the observed surface from its occluding contours where the camera motion is known. They introduced direct regularization on the 3–D surface to be reconstructed instead of smoothing the contours in the 2–D images. This is based on regularized bicubic B-splines. Also, they proposed to globally recover surface shape from small local patches. They fit patches separately and then stitch them smoothly together. Han and Medioni [53] proposed to recover general free-form surfaces from sparse range images using triangular B-splines defined over the meshes of arbitrary topology. The overall surface, they use, is $C^1$with preserved discontinuity at edges and junctions, meaning that surface smoothness can be changed on the places where data show presence of sharp edges or junctions. This is important because input data are pre-segmented into three dense potential fields: the surfaces, edges and junctions using global voting method of the same authors. Using triangular B-splines seems to offer generic way of coupling triangulations with smooth surface representation. In this case each B-spline patch is defined on the

domain of one facet, and is controlled with more control points then the three vertices of triangle. To stitch all patches together control points along the stitches are usually duplicated additionally increasing total number of control points. This may be impractical in case of complex and high resolution meshes where burst in model parameters will overshot number of mesh vertices. Also stays problem of maintaining continuity using strong regularization. Sullivan and Ponce [109] used also triangular B-splines for shape recovery from several registered photographs. The main source of information, they used, are the occluding contours. Their splines are created from the arbitrary triangulation with three spline patches per mesh facet. This additionally increases number of parameters to optimize.

Methods presented in [38, 57] are used to reconstruct surfaces from relatively clean laser-scanner data. They are not based on the predefined models, but they automatically retrieve structure from unorganized sets of data. However, when dealing with very noisy and incomplete data such as the stereo data that we intend to use, they would fail. They require initial surface polyginalization that is necessary for initial creation of the B-spline patch network, that is rather hard to retrieve from noisy data. A generalization of this work is done by Stoddart and Baker [108] where they used generalized bicubic B-spline surface. Their method is known as "slime" because it can fit arbitrary topology surfaces with locally adaptive meshing. The algorithm has low computational complexity and can adaptively generate control points of variable density in order to describe surfaces that are very detailed in some places and very smooth in some others. However, the slime method is designed to work well with optimal data sets where there is almost no noise and there is no gaps in the data. Initially it retries structure from the data using volumetric field function and the marching cube algorithm that generated triangular mesh. This mesh is further adapted by proposed seeding algorithm that produces initial control mesh.

Besides usage of B-spline patch networks for surface reconstruction, there have been efforts to use a single surface patch for shape recovery. Shen and Spann [103] addressed this problem by using bicubic B-spline surface. They actually used strategy of converting the surface estimation into curve estimation. In their experiments object surface is a closed surface topological to a sphere. There are several hierarchical levels of surface representation. Fitting is done from the coarse to the dense one, each time comparing change in parameters between representations. When movement of control points between consecutive hierarchical levels is small enough the result is obtained. Sengupta et al. estimate shape from a given set of depth maps computed from frame pairs in a video sequence. They use a bicubic B-spline surface with unknown number of knots. The optimization procedure recovers minimal number of spline basis functions that best fit given data.

2.1.1.3 Generalized Cylinders

Definition A generalized cylinder (hereafter GC) is a solid defined by its axis, cross-section curve and scaling function. They are flexible, special class of parametric shapes capable of modeling many real-world objects. We begin our discussion of GCs with its formal definition:

Definition 2.1.1. The generalized cylinder is the solid generated by a planar cross-section curve as it is moved and deformed along an asis.

This definition is quite general and, in practice, there have not been algorithms for shape recovery of general class of CG. *Straight homogeneous generalized cylinders* (hereafter SHGC) are a strict subclass of GCs and are defined as follows:

Definition 2.1.2. An SHGC is a GC with the following restrictions:

1. the axis is straight;
2. the cross-section curve is a simple, smooth C^2 curve orthogonal to the axis;
3. the cross-section are deformed only by scaling;
4. the scaling factor can be parametrized as a C^2 function of position along the axis.

If we associate to the axis of SHGC an orthogonal coordinate system $(O, \mathbf{i}, \mathbf{j}, \mathbf{k})$, where O is a point on the axis, and $(\mathbf{i}, \mathbf{j})$ is a vector basis of the reference cross-section's plane. The swept surface of a SHGC can be represented in the associated coordinate system by:

$$S(z, \theta) = \mathbf{OP}(z, \theta) = r(z)\rho(\theta)(\cos\theta\mathbf{i} + \sin\theta\mathbf{j}) + z\mathbf{k},\ (z, \theta) \in [a, b] \times [0, 2\pi] \quad (2.9)$$

The function $\rho(\theta)$ define the reference cross-section, while the function $r(z)$ defines scaling sweeping rule of the SHGC. For a constant value of parameter θ the curves on SHCG are called meridians, while for constant z they are called parallels.

Properties GCs and its special subclasses, like SHGCs and surfaces of revolution, are strictly-defined parametric surfaces. In theory, GCs can model a large variety of objects, but in practice it has never been used in this generic form. For shape recovery either SHGC or surfaces of revolution were used. Its flexibility for modeling objects of arbitrary geometric forms is rather limited. The shapes they can model are restricted to the rigid cylindrical forms. There is no way to meaningfully deform them out of the scope of their parameters. Local control of their shape cannot be achieved. As all other parametric surfaces they are easy to turn into meshes and render afterwards. SHGCs are C^2 differentiable as long as functions q and r are C^2 differentiable. As we will discuss later SHGC are fitted mainly to the image contours and range data what excludes noisy data we are interested in.

Literature Overview The invariant properties of the generalized cylinders and their silhouettes had been studied by various researchers [92, 58, 129, 78], and exploited for object recognition, reconstruction and object pose estimation. Substantial amount of work had been performed on shape recovery of GCs from image contours [97, 71, 58, 92]. Besides that reconstruction from 3–D data [35, 71, 123], such as range images was also present in the literature. Certain amount of work concentrated on shape and pose recovery from single monocular image [124, 129, 92].

2.1.1.4 Subdivision Surfaces

Definition Surface generation methods are important topic in Computer Graphics and CAD design. Much of the important work to date has concentrated on surfaces like Bézier and B-spline curves and surfaces represented by closed-form mathematical expression. However, these methods require huge number of surface patches in order to model objects of complex geometry.

A new set of methods, which utilize a meshes of polygonal shapes or a sequence of meshes to describe a surface, is now becoming popular. By doing this, freedom from the closed-form mathematical expression is achieved, and a wide variety of surface types can be expressed. The surfaces are commonly called subdivision surfaces as they are based upon the binary subdivision of the uniform B-spline curve/surface. In general, they are defined by a initial polygonal mesh, along with a subdivision (or refinement) operation which, given a polygonal mesh, will generate a new mesh that has a greater number of polygonal elements, and is "closer" to some resulting surface. By repetitively applying the subdivision procedure to the initial mesh, we generate a sequence of meshes that converges to a resulting surface.

As it turns out, this is a well known process when the mesh has a "rectangular" structure and the subdivision procedure is an extension of binary subdivision for uniform B-spline surfaces. Therefore, we first present a somewhat extensive study of the uniform B-spline case, and then show how these results can be generalized to treat the case when the mesh is not based on a rectangular structure.

We can represent uniform biquadratic spline surface in a matrix form as follows:

$$P(u,v) = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} BPB^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}, \; P = \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} \\ P_{1,0} & P_{1,1} & P_{1,2} \\ P_{2,0} & P_{2,1} & P_{2,2} \end{bmatrix} \tag{2.10}$$

where outer parts of vectors and matrices, $\begin{bmatrix} 1 & u & u^2 \end{bmatrix} B$ and $B^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}$ create the quadratic uniform B-spline blending functions, while matrix P defines the geometry of the patch, that is in this case 3x3 control polygon with rectangular faces. If we introduce a subdivision rule, represented by the refinement matrix S, that subdivides control polygon P creating a new control polygon $P^{'} = SPS^T$, we can represent a new subdivided surface that is closer to the limit biquadratic B-spline surface as follows:

$$P^{'}(u,v) = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} BP'B^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} BSPS^TB^T \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix} \tag{2.11}$$

Note that each new control point $P^{'}_{i,j}$ is generated according to the four points on the face it belongs to. The subdivision schema can be simply specified by using *subdivision mask,*

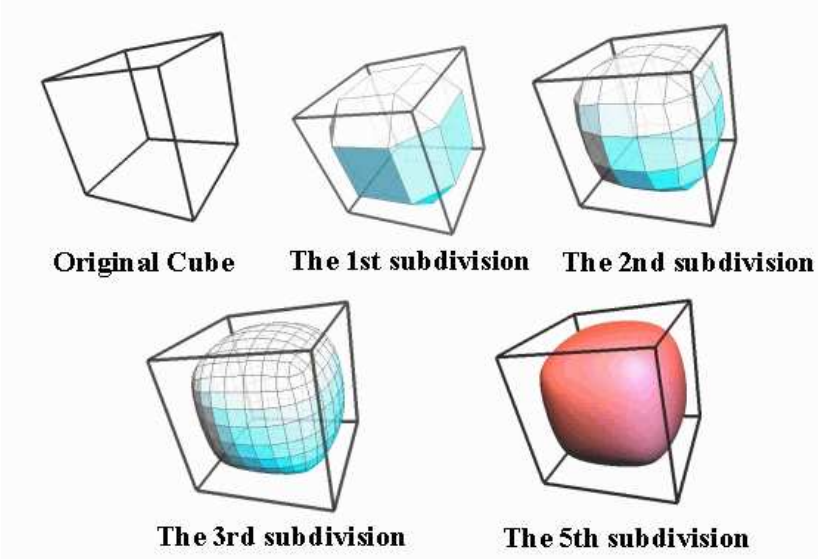


Figure 2.3: Doo-Sabin's subdivision schema applied to the cube. The images are found on the web and are courtesy of Zheng XU [127].

which specify the ratios of the points on a face to generate the new points. In this case, the subdivision masks are as follows:

$$\begin{array}{cccccccccc} 9 & \_ & 3 & \quad 3 & \_ & 1 & \quad 1 & \_ & 3 & \quad 3 & \_ & 9 \\ | & & | & \quad | & & | & \quad | & & | & \quad | & & | \\ 3 & \_ & 1 & \quad 9 & \_ & 3 & \quad 3 & \_ & 9 & \quad 1 & \_ & 3 \end{array} \tag{2.12}$$

These algorithms are from a class called "corner cutting" algorithms - that is, their action can be described by cutting the corners off of polygonal meshes. The study of quadratic and cubic B-spline surfaces led respectively to the two better known surface subdivision schemes, the Doo-Sabin [36] and Catmull-Clark [16] methods. It is important to mentioned a method by Charles Loop [79] that is based upon a mesh with a triangular based structure. Doo-Sabin schema relays on the subdivision principle mentioned above, that is based on bi-quadratic uniform B-spline surface subdivision and can deal with meshes of arbitrary topology. In Fig. 2.3, it show how it behaves when subdividing a cube. Cutmall-Clark's schema is an extension of the Doo-Sabin's work and is based on bi-cubic uniform B-spline surface subdivision, that led to a better subdivision, as shown in Fig. 2.4.

Note that the Cutmull-Clark subdivision surface produces smoother corners compared to the Doo-Sabin surfaces. This is because of using higher order B-splines as basis for the refinement process. Finally, in Fig. 2.5 we depict Loop's [79] subdivision schema, which is designed for the triangular meshes.

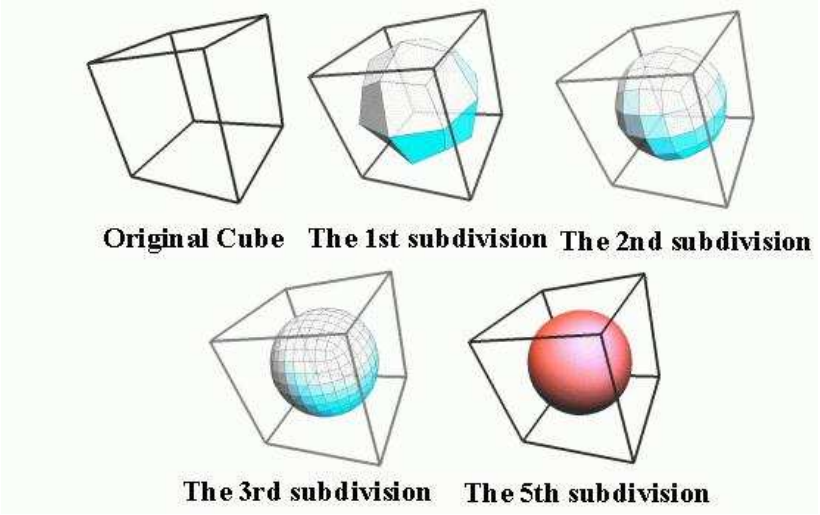


Figure 2.4: Cutmall-Clark's subdivision schema applied to the cube. The images are found on the web and are courtesy of Zheng XU [127].

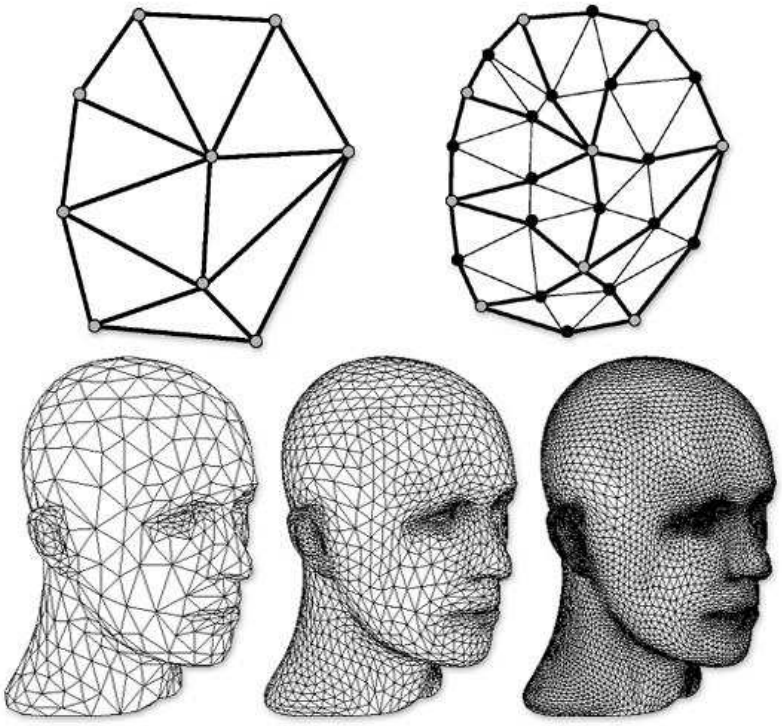


Figure 2.5: Loop's subdivision schema. Top row: Subdivision of the piece of mesh. Bottom row: Subdivision of the generic human face model. The images are found on the web and are courtesy of Zheng XU [127].

Properties Subdivision surfaces appear as extension of B-spline surfaces. However, they lost their closed-form mathematical formulation for the price of arbitrary mesh modeling. The idea to start from the mesh of arbitrary geometry and topology and smoothing it by subdividing that initial mesh is very attractive from the modeling point of view. However, the limit surface cannot be expressed directly as a function of initial control mesh. When it comes to rendering and fitting, the subdivision surfaces can be considered exactly as any other polygonal mesh. The main difference is that number of parameters sufficient to control the shape of the mesh is equal to the number of control points of the initial mesh (control mesh) that is subsequently divided. In practice, computing the distance function, involves projecting the observation points onto the subdivision surface. Since this is not feasible, as the surface is defined only as a limit of an infinite process of subdivision, the observations are projected onto the piecewise approximation of the limit surface obtained by subdividing r times initial surface. Points of this approximated surface can be expressed as a linear combination of the initial surface vertices using the subdivision rules. Again, as in a case of polygonal meshes, we have non-differentiable distance function, that is not ideal, especially for fitting 2–D occluding contours.

Literature Overview In last decade subdivision surfaces became very popular in Computer Graphics. However Computer Vision community did not show much interest in those surface representations. Among first researchers that used subdivision surfaces for automated fitting to the unorganized laser-scanned data were Hugues Hoppe et al. [57], that extended Loop's subdivision schema so to model shape features, such as edges and creases. Their algorithm consists of three phases: estimation of the objects geometry and topology from unorganized data, optimization of the retrieved mesh structure and finally using this mesh as a control mesh for fitting its subdivided version to the unorganized data. The last phase is the one we are particularly concerned, and the optimization is performed by varying number of control mesh vertices (initial mesh extracted from the data), their connectivity, their position and the number and locations of sharp features. Distance function is obtained by attaching laser scan data points to the closest mesh facet obtained by subdividing the initial control mesh several times. Laser scan data are relatively noise free, so particular regularization energy is not used in this work. Fitting of Cutmull-Clark subdivision surface to a given shape within a prescribed tolerance is presented by Nathan Litke et al. [77]. Instead of using classical least-square fitting they used method of quasi-interpolation [25] that offers fast and local fitting solution. However, authors did not address problem of establishing correspondences between the data points and the generic models. Since the data were acquired by cylindrical scan, they use simple cylindrical coordinates for the correspondences. Quasi-interpolation allows to infer new control points positions from the data. Also, in this work each subdivided version of the initial control mesh is fitted to the data and the process is finished when vertices of two successive subdivided meshes do not differ more then the given tolerance. This is actually hierarchical fitting of the subdivision surface, where each subdivided surface is considered as one item in the hierarchy of meshes going from the coarse to fine ones. Scheib et al. [98] presented a method of fitting and render-

ing of large scattered data using subdivision surfaces. They applied their technique to the therein reconstruction, where the initial control mesh is actually planar surface with associated height. Their representation is adaptive triangulation obtained by binary triangle tree hierarchy. They actually fitted locally to each data point a polynomial of degree less then three. For each vertex point of subdivision surface they associated height that is obtained by evaluating the value of the polynomial function at that vertex. Fitting is then performed locally by using SVD. Displaced subdivision surfaces proposed by A. Lee, H Moreton and H. Hoppe [73] are interesting surface representation that offers compact solution for detailed surfaces. The idea is to start from the given high resolution irregular mesh, that is hard to manipulate, and represent it as a subdivision surface with associated displacement map. The initial control mesh is very compact, its limit surface is called domain surface and represents smooth approximation of the input mesh, and displacement map consists of scalar values that denote how much each sample point on the domain surface is far from the fine details on the input mesh. Fitting such surfaces from unorganized data points was proposed by W.K.Jeong and C.H. Kim [66]. They compute the control mesh from the bounding box that encloses the data, by subdividing it and shrink-wrapping it to the points with point-based simplification algorithm whose error metric is the distance from the point to the mesh. After that the algorithm is similar to one of original paper of displaced subdivision surfaces [73]. Also, this algorithm requires data without any noise. Another paper [18] of fitting subdivision surfaces to the unorganized points introduced different distance metric. Instead of using point distance minimization they used more complex distance function called surface distance. This metric requires estimation of the normals and curvatures of the data points what is not easy to do in case of the noisy data.

2.1.2 Implicit Surface Models

Implicit surfaces, as their name suggests, are surfaces that are not given explicitly. Instead, they are given in closed mathematical form by a single equation. For example the equation for a sphere: $x^2 + y^2 + z^2 - r^2 = 0$ describes infinite number of (x, y, z) points at the distance r that lie on the common surface. If we take any other point in space, plug it in the above equation, we will obtain some non-zero value. The returned value can be used to determine whether the point is inside or outside an implicit surface. The equation of the sphere can be rewritten as:

$$F(x, y, z) - T = 0 \tag{2.13}$$

Now, by changing the value of constant T we obtain different surfaces (depending on function F) that can be visualized by collecting all those (x, y, z) points where the equation Eq. 2.13 gives the value zero. Each such surface is called iso-surface and the function F is called scalar field function, since it always returns scalar value.

There are three different ways to formulate implicit surfaces depending on the way of its construction. First we will address implicit surfaces composed of simple volumetric primitives. Then algebraic implicit patches similar to the parametric B-spline patches will

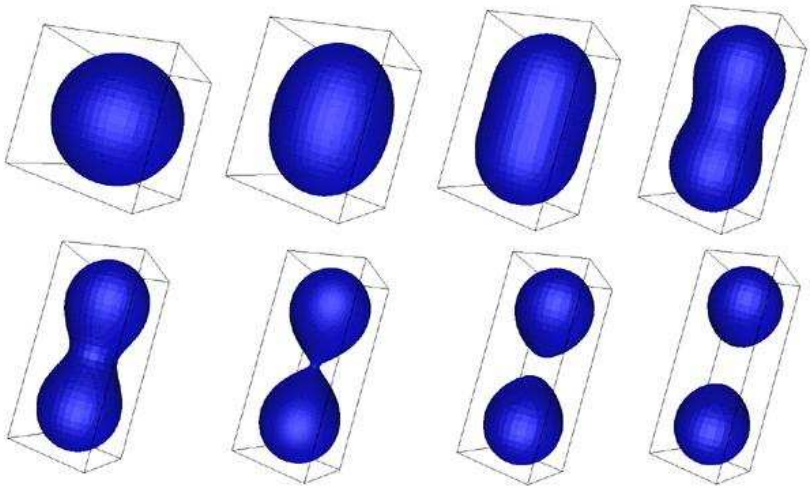


Figure 2.6: Implicit surface made of volumetric primitives. When two primitives coincide the overall scalar field results in a bigger sphere as shown in a first image. Further, separating those two primitives the range of their influences less and less overlap, until finally they become two separate primitives. The images are found on the web and are courtesy of Paul Bruke [15].

be discussed. Finally, the family of variational based surfaces, also known as a thin plate splines, will be described.

2.1.2.1 Volumetric Primitives

Definition There are a number of approaches to modeling using implicit surfaces. We can categorize them into two groups: *simple primitives* and *skeletal primitives*. The first category, simple primitives, refers to implicit modeling techniques that build the implicit primitives by a scalar field around a single point creating volumetric primitives like spheres and ellipsoids as depicted in Fig. 2.6. The skeletal primitives can create implicit surface around more complex geometric primitives such as line segments and polygons as shown in Fig. 2.8. The second one can be considered as a generalization of the first one. Basic simple primitives, known in a literature are *Blobby Molecules [8]*, *Metaballs [39]* and *Soft Objects [125].* They are all very similar, but their main difference is in using different scalar field function as shown in Fig. 2.7 and discussed bellow.

For *Blobby Molecules*, Blinn uses Gaussian density function around a single atom (based on behavior of hydrogen atoms):

$$D(\mathbf{X}) = b \exp^{-ar^2} \tag{2.14}$$

where r is a distance of the point $\mathbf{X} = (x, y, z)$ to the center of the atom. For multiple atoms the overall density at the given point $\mathbf{X}$ is calculated by summation of all densities for all atoms:

$$D(\mathbf{X}) = \sum_{i=0}^{N} b_i \exp^{-a_i r_i^2} = T \tag{2.15}$$

where r_i is distance from the point $\mathbf{X}$ to the center of i-th atom. The blobbiness of the model can be controlled by changing parameters a_i and b_i. The overall surface is defined where all the points have value equal to some threshold T.

For *Metaballs*, the density function is piecewise polynomial, whose influence is not infinite, as it was for exponential one of blobby molecules, and is limited to the size of the metaball. It is given as follows:

$$w(\mathbf{X}) = \begin{cases} d_i(1 - 3(\frac{r_i}{b_i})^2) & 0 \leq r_i \leq \frac{b_i}{3} \\ \frac{3d_i}{2}(1 - \frac{r_i}{b_i})^2 & \frac{b_i}{3} \leq r_i \leq b_i \\ 0 & b_i \leq r_i \end{cases} \tag{2.16}$$

where r_i is a distance of the point $\mathbf{X}$ from the center of i-th metaball, d_i is the weight of i-th metaball and b_i is the radius of i-th metaball. Similarly for multiple metaballs the field potentials are added as in Eq. 2.15.

Soft Objects replaced piecewise polynomial density function by simple polynomial. Additionally, the scalar field value is truncated to zero at a certain distance (radius of influence):

$$C(r) = \begin{cases} a(1 - \frac{4}{9}\frac{r^6}{b^6} + \frac{17}{9}\frac{r^4}{b^4} - \frac{22}{9}\frac{r^2}{b^2}) & r \leq b \\ 0 & r \geq b \end{cases} \tag{2.17}$$

where r is the distance from $\mathbf{X}$ to the considered key-point and b is the radius of influence of the key-point, while a is a scaling factor. Several primitives are blended together by summing their contributions in the areas of influence as it was done in Eq. 2.15.

The skeletal primitives extend concept of density function that was computed from a single point. Here, field potential is computed from more complex geometric (skeletal) primitives such as lines and polygons. For this purpose the distance function that computed distance of points in space from the primitive has to be employed. In [9] Bloomenthal proposed two approaches to this: *distance surfaces*, where the potential is calculated from the distance of the nearest point on the skeleton and *convolution surfaces* where it is found from all points on the skeleton by integration.

Distance surfaces employ following scalar potential field function:

$$f(S, \mathbf{x}) = \max_{s \in S} \exp\left(\frac{-\|s - \mathbf{x}\|^2}{2}\right) \tag{2.18}$$

where S is the skeleton, $\mathbf{x}$ is the point for which the potential is calculated and s is a point on the skeleton. This function gives the union of the volumes generated by all the

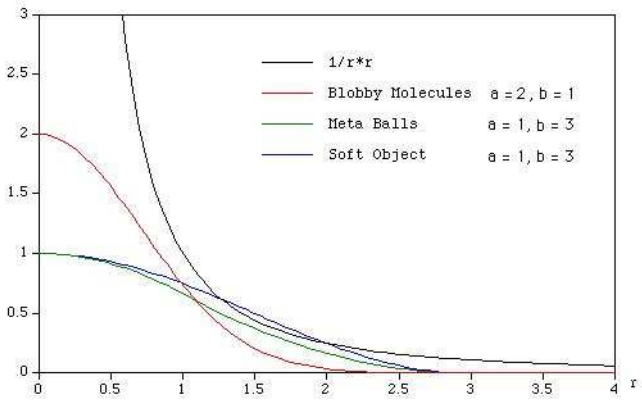


Figure 2.7: Scalar potential filed functions of Blobby Molecules, Metaballs and Soft Objects. Implicit surface creation on the left is made using simple potential filed that is inversely proportional to the square of the distance, $1/r^2$, to the center of the primitives.

individual points on the collective skeleton, S. Disadvantages of this formulation are: non-differentiability obtained by introducing $\max$ function and that bulges and creases may occur in the area where the skeletons of non-convex objects meet.

Convolution surfaces are proposed in the same work of Bloomenthal as a solution to possible unwanted bulges and creases at the junctions of skeletal elements. A point $\mathbf{x}$ in space has following potential field value computed from all points of the skeleton by integration:

$$f(S, \mathbf{x}) = \int_S \exp\left(\frac{-\|s - \mathbf{x}\|^2}{2}\right) ds \tag{2.19}$$

where S is the skeleton, $\mathbf{x}$ is the point for which the potential is calculated and s is a point on the skeleton. However, convolution surfaces lose their analytical representation for the resulting surface, since the integral cannot be solved analytically, and alternative methods has to be proposed as it is done in [9].

Properties Implicit surfaces guarantee a continuous and smooth surface that has compact mathematical representation given by a single formula such as the one of Eq. 2.13. This representation allows direct minimization of the algebraic distance that is differentiable over the parameters defining its shape and position in space. Having such properties, implicit surfaces are ideal for automated fitting to the noisy data. By construction, the limited range of influence of the potential field function eliminates outliers. However, it is hard to create arbitrary shaped objects of the complex geometry using only simple primitives, such as metaballs, blobby molecules or soft objects. Some complex objects can be cre-

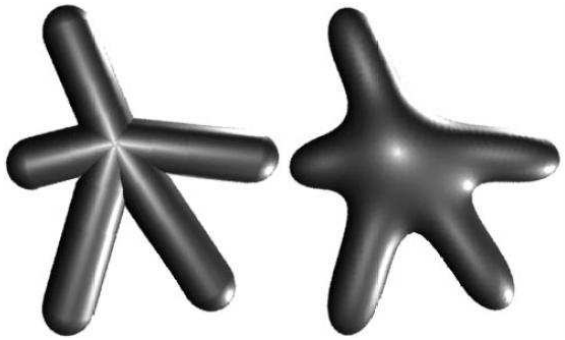


Figure 2.8: Distance and convolution surface around star like skeleton made of line segments. The images are taken from [10].

ated using CSG and ability to easily apply boolean operation to their scalar potential fields. Skeletal primitives are more efficient for creating arbitrary shaped objects, but because of non-differentiability of distance surfaces and non-analytical form of the convolution surfaces they were not suitable for automated fitting. In general, implicit surfaces are hard to deform in free form fashion and only well established operations, such as, bending, twisting and tapering are easy to make. Visualization of implicit surfaces is an expensive operation, since the surface has to be either turned into the polygonal representation or rendered by ray tracing. Physically based surface deformation and animation are also well known good properties of implicit surfaces. During the animation there are two problems that may appear: coherence loss happens when two implicit primitives are disconnected by placing them too far from each other, and unwanted blending, that is characteristic for creating bulges at the junction of two closely put primitives.

Literature Overview Implicit surfaces were first introduced to the Computer Graphics by Blinn [8], whose *Blobby Molecules* were used to visualize electron density field. This work was followed by Nishimura *[39]* and G. Wyvill *[125]* with their *Metaballs* and *Soft Objects* where the scalar potential field function had been changed. Further in graphics, many researches addressed problems of interactive modeling with implicit surfaces. We can recommend an excellent book on implicit surfaces by J. Bloomenthal [12], that gives complete overview on creating, modeling, deformation and rendering of implicit surfaces.

In Computer Vision, implicit surfaces were extensively used for reconstruction and tracking of deformable and articulated objects, such as hand and human body. Main ingredient in those models are quadrics and superquadrics, which have both parametric and implicit representation. Its implicit representation, as already reviewed in Section 2.1.2, demonstrated good fitting properties for shape recovery [51, 106, 94, 86, 76]. Fitting dynamic, physically based implicit object took also interest by the researchers [99, 83, 28, 115, 117, 114]. Recently, Dewaele and Horaud [32] proposed to use complex 3D hand model made of ellipsoids. They used separate ellipsoids attached to each part of the hand skeleton to track their

rigid motion using well known iterative closest point algorithm(ICP) in its EM version. The algorithm was adapted for articulated and deformable object, followed by another adaptation called EM-ICPS, where point to point and point to surface distances are taken into account. The problem of drift that might happen during tracking is solved using appropriate implicit surface model created by blending separate ellipsoids attached to hand skeleton parts. The problem of interest points displacents caused by skin deformation is solved by introducing skinning technique taken from computer animation in order to compute motion of the points on the skin.

We would like to distinguish work of Sullivan and Ponce [110] that addresses the problem of automatic construction of implicit surface models from set of 2–D and 3–D images and uses this model for pose computation, motion and deformation estimation, and object recognition. They minimize the mean-squared geometric distance between a set of points or silhouette rays and implicit surface model. In modeling, the state vector is composed of unknown surface parameters, while in pose and deformation it consists of global rotation and translation vectors. Another interesting application of implicit surfaces is done by Plaenker and Fua [91], where articulated body model with attached metaballs is used for human body reconstruction and tracking from stereo and silhouettes.

An interesting work of Bajaj [1, 2, 3] addresses a modeling problem with algebraic implicit patches. He used a Bernstein-Bézier form of trivariate polynomial within thetahedron, such that the real zero contour of the polynomial defines a smooth algebraic surface patch. Such patch they call A-patch, and is used to interpolate arbitrary surface triangulation or fit unorganized data. If unorganized points are fitted, the Delaunay triangulation is used to divide the data points into groups delineated by tetrahedron. There are two primary steps in the algorithm: (1) creating the tetrahedron attached to each facet, edge and vertex of triangulation, and (2) obtaining the coefficients of the Berstain polynomial that approximate enclosed facet or data. C^1continuity is preserved between patches by forcing several coefficients of each polynomial to be equivalent between neighboring patches. The surface constructed in this way is a piecewise algebraic implicit surface and it losses compact characteristics of global representation, that ordinary implicit surfaces posses. The operations, such as, collision detection, morphing, blending, and modeling with constructive solid geometry become more difficult to perform since the representation is no longer a single analytical function.

2.1.2.2 Variational Based Surfaces

Definition Given a set of 3–D data points $X = \{x_i\}_{i=1,N} \subset R^3$ through which the surface should pass, and also identifying interior and exterior surface points it is possible to define a 3–D implicit function $f(x, y, z) = 0$ that interpolates those surface points. The problem defined in this way is actually scattered data interpolation problem. The isosurface of such implicit function is called *variational implicit surface* according to Turk and O'Brien [119]. Implicit surface created this way is *radial basis function* (RBF) that appeared to be the smoothest interpolant that minimizes the energy function measuring an aggregate squared curvature of the implicit function over its region of interest. In general,

an RBF is a function of the form:

$$f(x, y, x) = p(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{2.20}$$

where $p(\mathbf{x})$ is a polynomial of low degree, the coefficients λ_i are real numbers, the *basis function* ϕ is a real valued function on $[0, \infty)$, usually unbounded, and $\mathbf{x}_i$ are data points usually called RBF centers. Popular choices for the basis function include the thin-plate spline $\phi(r) = r^2 \log(r)$ (for fitting smooth functions of two variables), the Gaussian $\phi(r) = exp(-cr^2)$ (mainly for neural networks) and the multiquadratic $\phi(r) = \sqrt{(r^2 + c^2)}$ (for various applications in particular for fitting topological data). For fitting functions of three variables the good choices include biharmonic $\phi(r) = r$ and triharmonic $\phi(r) = r^3$ splines. Everywhere $r = \|\mathbf{x} - \mathbf{x}_i\|$ is Euclidean distance of the point in space $\mathbf{x}$ to the RBF center $\mathbf{x}_i$.

Properties RBFs have excellent interpolation properties and can turn cloud of scattered data points into a single implicit function capturing the tiniest details. This in turn allows conversion of arbitrary polygonal mesh into implicit surface. To overcome the trivial solution of scattered interpolation problem, besides x_i data points, one have to provide, so called off-surface points defining interior and exterior of the resulting surface. This can be done by adding new RBF centers at the certain distance on the both sides of the estimated point normals. This additionally increases number of equations to solve and in a worst case it is possible to end up with $3N$ equations, where N is initial number of data points, i.e RBF centers. Finally, the resulting surface will be defined by $3N$ λ coefficients. Surface obtained in this way depends uniquely on positions of RBF centers and its shape cannot be changed intuitively, since λ coefficients do not have any geometric meaning. Deformation of such surfaces is almost impossible to perform, and if the position of data points is changed the only way to obtain the new shape is by interpolating them again. If one wants to perform fitting of surfaces represented in this way to the noisy data, the final result will interpolate noise as the relevant information.

Literature Overview Variational implicit surfaces are studied by the Computer Graphics researchers that wanted to interpolate unorganized scattered data points form laser scanners and produce compactly represented surface which can be than used for re-meshing. Turk and O'Brien used variational implicit surfaces [120, 118] for shape reconstruction and shape transformation. Traditionally, shape transformation using implicit functions is performed in two distinct steps: 1) creating two implicit functions, and 2) interpolating between these two functions. They combine these two tasks into a single step. They create a transformation between two N-dimensional objects by solving a variational problem in N+1 dimensions. For the case of 2–D shapes, they place all data constraints within two planes, one for each shape. These planes are placed parallel to one another in 3–D. We then create a variational implicit surface in 3D from these constraints. Intermediate shapes are simply

the zero-valued contours of 2–D slices through this 3–D function. Shape transformation between 3–D shapes can be performed similarly by solving a 4–D interpolation problem. The transformations produced by this method appear smooth and natural, even between objects of differing topologies. This allows to perform effective morphing between the object of completely different geometries. As we already said, direct solving of this interpolation problem requires lots of memory and time for obtaining its solution and for problems where there are more there 2000 data points becomes impractical. Carr et al. [40, 41, 64] proposed to solve this problem by using Fast Multiple Methods(FMM) [6]. They managed to interpolate huge amount of data involving millions of points in a reasonable time. They recently addressed problem of noisy data where the smoothing term was introduced and the resulting surface is smoothed in the phase of conversion of implicit surface to the polygonal mesh. This is not applicable to the stereo data where noise is much larger then the one produced by the scanners.

2.2 Tracking of Deformable Objects

Motion of physical object in the world is generally nonrigid. In Computer Vision there is a growing interest in recovering the motion of nonrigid or deformable objects. The deformable models can be categorized according to the their way and degree of deformation in three groups: articulated, elastic and fluid. Articulated models are composed of rigid parts associated to the skeleton. Even thought the motion of the rigid parts is not deformable the overall motion is nonrigid. This kind of models was widely used for creating articulated models of the human body used for tracking and motion analysis. Elastic models are designed to perform nonrigid motion that confirms to the certain degree of continuity and smoothness. Fluid motion violates even the continuity assumption and may involve topological variations and turbulence deformations.

In this work we are particularly interested in the elastic models and their use for recovery of the nonrigid motion of deformable objects. The applications of elastic motion, for example, relate to facial reconstruction, animation of nonrigid objects from images and their use in augmented reality applications, clinical examination of the heart and soft tissues, and model based image compression. Most of the approaches that involve nonrigid motion assume an object model and try to model deformation as the variation of the model's parameters. This approach has advantage of constraining the degree of freedom exhibited by the deformable objects. The task of motion recovery is often reduced to the problem of parameter estimation of the deformable model. We can distinguish two types of deformable modes: parametric models and physically based models. This two deformable model types will be overviewed in the following subsections.

2.2.1 Parametric Models

Parametric models are suitable for defining the global shape of the objects and a priori knowledge of the object must be provided. Consequently, most parametric models are able

to model limited class of objects and without proper modification are not ideal for modeling of dynamically deformable objects. Throughout the review on the surface representation, we already mentioned all sorts of surface models. All of them, including explicit and implicit representations, are expressed in terms of certain parameters that control their shape and are used for static shape recovery. We are going to mention those works that were used for dynamic shape recovery.

One of often used deformable models is model of the human face. Human face model is represented as polygonal mesh used for face reconstruction and animation. Capturing of the face dynamics is an example of motion recovery of the deformable objects. There is a work of Blanz and Vetter [7], on animating faces in images and videos. In order to animate novel faces the deformable face model is build such that it captures common representations of different faces and facial expressions in a vector space of 3–D shapes and textures. This space is computed from 3–D scans of neutral faces and scans of facial expressions. DeCarlo and Metaxas [27] build a dynamic face model that when coupled with optical flow and edge data produces facial deformation. Physical models of faces have been proposed for analysis of the facial motion as they allow for more degrees of freedom. Modeling is mainly focused on solving the tracking of canonical facial expressions.

Reconstruction and tracking of the elastic motion of the heart's left ventricle in [4] is another example of deformable parametric model. This time model is represented as a superellipsoid. Its shape is additionally parametrized by free form deformation (FFD) control points which are placed on the rectangular grid around the superelipsoid. Solina and Bajcsy [106] recover the superquadrics with global deformations from range images.

In several papers Bookstein [13, 14] illustrated the potential applications of thin-plate splines for modeling of biological shape changes, production of biomedical atlases and image feature extraction.He demonstrated the decomposition of deformations by principal warps, which are geometrically independent, affine-free deformations of progressively smaller scales. Terzopoulos [52] defined dynamic NURBS as a physics-based framework for geometric design.

Implicit surfaces were used mainly for modeling in Computer Vision [110, 90]. The paper of Desbrun and Cani [31] presents a hybrid model for animation of soft inelastic substance which undergo topological changes, e.g. separation and fusion and which fit with the objects they are in contact with. The model uses a particle system coated with a smooth iso-surface that is used for performing collision detection, precise contact modeling and integration of response forces. The animation technique solves those three problems inherent in implicit modeling.

2.2.2 Physically Based Models

Physically based models are fundamentally dynamic and are governed by the lows of rigid and nonrigid dynamics expressed trough a set of Lagrangian motion equations. A number of physically based models have been developed for image analysis, including snakes, symmetry-seeking models, deformable superquadrics, deformable templates, and modal models.

The snake model is perhaps the most well known and widely used physically based model. A great deal of attention has been devoted to the extension of the named prototype first proposed by Kass et al. [70]. The original snake model is a class of active contours that evolve under the influence of external potentials but are constrained by internal energies. When augmented by Lagrangian mechanics, dynamic snakes with intuitive physical behaviors are developed [113]. It is not surprising that the snake model can also be generalized to deal with 3D images, as shown by Cohen et al. in [20, 21]. This newly formed class of deformable surfaces, called balloons, is able to conform to image features and external forces in a way similar to the original snake model. An evolution equation similar to that in dynamic snakes has also been formulated. Cohen et al. have successfully applied this model to the segmentation of 3D MRI images as well as to establishing correspondence between a deformable surface and an anatomical atlas. A more recent extension to snake models can be found in [112], where the essential elements of physically based models and probabilistic approaches are incorporated. A Bayesian framework is introduced and the original energy-minimizing problem is transformed to an MAP problem. To further exploit the power of probabilistic modeling, Szeliski et al. [112] have developed a sequential estimation algorithm using the Kalman filter. Known as the Kalman snake, this dynamic system is able to integrate nonstationary, noisy observations over time. It provides the flexibility to design behaviors that may not be possible with purely physically based models. Moreover, model parameters can be derived from statistical models of sensors, rather than chosen heuristically.

The free-form deformable surface model proposed by Delingette et al. [29, 30] is conceptually similar to the active contour model [70]. They model an object as a closed surface that is deformed subject to attractive fields generated by input data points and features. A fundamental conflict in shape representation is that a modeling primitive should be general enough to handle a wide variety of scenes, yet simple enough to be usable for tasks such as recognition and manipulation. To balance these conflicting requirements, the authors suggested a coarse/fine approach where features affect the global shape while data points control its local shape.

Deformable templates, such as those employed by Yuille et al. [128] to extract facial features, mark a blend of parametric representation and physical-based methods. The template is described by a parametrized geometrical model. The goodness of fit between the deformable model and the image is measured by the interaction energy, which contains contributions from various image features. Optimal fit is obtained when the energy is minimized.

Deformable superquadrics [114] are dynamic surface models with global and local deformation properties inherited from superquadrics ellipsoids and membrane splines. The combined local/global representation is aimed at solving the conflicting goals of shape reconstruction and recognition we addressed earlier. Additional deformational degrees of freedom are gained from the incorporation of global deformation such as tapering, twisting, and bending [23]. By casting the fitting of time-varying visual data into the Lagrangian mechanical framework, the equations of motion governing the behavior of the deformable su-

perquadrics can be developed. When augmented by Kalman filter theory [82], the dynamic system becomes a recursive shape and motion estimator which employs the Lagrange equation of dynamic surfaces as a system model. In [24], deformable superquadrics that combine Kalman filter with additional constraints are employed to track articulated objects.

Inspired by modal analysis in linear mechanical systems, Pentland [86] developed a system that is capable of automatically recovering deformable part models based on the finite element method (FEM). Nonrigid object behavior is described by modal dynamics, i.e., by the superposition of its natural strain or vibration modes. By limiting the number of modes used in the representation, the analysis of nonrigid motion can always be transformed to an overconstrained problem. Later, the same model combined with an extended Kalman filter is applied to recover nonrigid motion and structure from contour [87] as well as optical flow data [87]. A major limitation of the modal framework is that objects must be described in term of the modes of some prototype shape. Such a procedure implicitly imposes an a priori parameterization upon the sensor data. It is thus more suitable for modeling than for tracking purposes. To address this problem, Sclaroff and Pentland [99] recently developed a new method that computes the object s vibration modes directly from the image data. Nastar and Ayache [85] followed similar physics principles and developed elastic models for nonrigid motion tracking. The notable property of their model is that the governing dynamic equations are linear and decoupled for each coordinate, regardless of the amplitude of deformation. Algorithmic complexity is therefore significantly reduced.

2.3 Summary

In this chapter we gave the overview of the surface representations used in Computer Vision and Computer Graphics for the automated reconstruction from images or laser scans. In this thesis we are also interested in tracking of deformable objects, and also discussed surface representations in this context. We divided all surface representations in two main groups: explicit and implicit surface representations. We discussed their properties in terms of automated fitting and their deformability. Also, various applications of those surface representations for shape modeling, reconstruction and tracking are addressed and the literature overview has been given. In general it can be said that explicit surface representations are well suited for graphics purposes, including rendering, editing and animation, but less so for fitting and automated modeling, because of the non-differentiable distance functions and difficulty of handling silhouettes. The reverse can be said of implicit surface representations. In this thesis, we propose to combine the strengths of both approaches and to avoid their drawbacks. For that reason we created *implicit meshes*, explained in Chapter 4, which represent a robust surface representation that can be used for automated shape recovery from images and can efficiently take advantage of various data sources coming from images.

3 Generic Surface Parameterization

In the previous chapter we gave a wide overview of surface models used for reconstruction and modeling. Those models are usually controlled by their internal parameters, integrated inside of the model. They are directly responsible for its construction and shape modification. For example, polygonal mesh is defined by positions of its vertices, parametric models by its parameters, implicit surfaces by their volumetric primitive constants, subdivision and tensor product surfaces by their control points etc. The *surface parameterization* is a way of defining a surface shape and its deformations by a set of variable parameters. The already mentioned parameterizations are imposed by model construction and appear to be *internal*. However, besides this internal parameterizations there are the other ways of surface modification using the set of *externally* defined parameters. Usually, the surface model can be expressed completely by using this external set of parameters, and only manipulation of these parameters controls the existing shape and generates the new ones. Expressing the surface model in terms of those parameters, besides efficient shape modification, intends to dramatically decrease number of model's internal parameters. This is very important when automated fitting algorithms come to the interest.

In this chapter we advocate a parameterization that can be applied to the objects of any geometry and complexity, thus appears to be generic. It allows efficient shape deformation and significant dimensionality reduction. It is based on the *free form deformation (FFD)* technique, that basically defines a control structure made of 3D vertices, which controls the surface model. Surface model points are easily expressed in terms of control points (vertices). More specifically we will use special type of FFD that is called *Dirichlet Free Form Deformation (DFFD)*. It is both suitable for automated fitting required by Computer Vision algorithms and manual editing and animation required by Computer Graphics designers.

3.1 Dirichlet Free Form Deformations (DFFDs)

Free-form deformations (FFDs) constitute an important approach to geometric shape modification that has been extensively investigated for computer animation and geometric modeling [100, 22, 68, 17, 84]. In the vision community, they have also been used to fit parametric models to medical data [111, 4] or animation masks to semi-automatically extracted silhouette data [75]. That approach, however, takes silhouettes extracted from orthogonal images as input and does not allow for potential errors in the data. In this work, we show that FFDs are also very effective to fit deformable surface models to the kind of noisy 3–D data that vision algorithms such as stereo tend to produce.

The initial FFD approach [100] and all subsequent ones involve embedding the object

model into a volume whose shape can be changed by moving a number of control points. Here, we take the embedded object to be a triangulated mesh and the embedding guarantees that each model vertex is influenced by the deformation of the control grid. In the original FFD, the control points must be placed on a regular lattice, which severely limits the range of allowable deformations that can be modeled. Most FFD extensions aim at overcoming this limitation, often by using a more sophisticated interpolant, but without addressing the basic problem that there is little flexibility in the positioning of the control points.

By contrast, DFFDs [84] remove the requirement for regularly spaced control points that is the main conceptual geometric limitation of FFDs This is achieved by replacing the typical rectangular local coordinates by generalized natural neighbor coordinates, also known as Sibson coordinates[104] and using a generalized interpolant [42]. That property give us the ability to place control points at arbitrary locations—that is, on the object, inside of it or outside—rather than on a regular lattice, and thus much greater flexibility. In particular, some of the control points can be important feature points that must be controlled in a specific way. This idea comes from the data visualization community that relies on data interpolation and, thus, heavily depends on local coordinates.

We concentrate on upper body modeling but we will, however, argue that the approach is generic and can be applied to any task for which deformable facetized models exist. In particular, we will show that we can also use our approach for high-resolution modeling of the human ear.

3.1.1 Sibson Coordinates

DFFD can be viewed as a data interpolation scheme where the interpolating function is a 3–D function specifying point displacements. The displacement is known at the control points and we want to interpolate it to the points of the object to deform. To this end, DFFD replace standard rectangular or barycentric coordinates that constrain the control grid's shape by Sibson coordinates [104]. More precisely, given a set of control points, every vertex of the explicit mesh is influenced by a subset of those control points. The Sibson coordinates depicted by Fig. 3.1 quantify those influences and are computed as follows.

Let $Q = \{P_1, \ldots, P_N\} \in R^3$ be the set of all control points whose Delaunay triangulation and Voronoi diagram we compute. Let p be a triangulation vertex and $Q_p = \{P_k\}_{1 \leq k \leq N_p}$ be the subset of control points whose circumscribed spheres contain p, as shown in Fig. 3.1(b). The elements of Q_p are the natural neighbors of p. Their relative influences are obtained by computing the Voronoi diagram of the augmented set $Q'_p = \{p, P_1, \ldots, P_{N_p}\}$ depicted by Fig. 3.1(d) and taking the Sibson coordinate u_i of vertex P_i to be

$$u_i = \frac{Vol(P_i) - Vol'_p(P_i)}{Vol'_p(p)} \quad , \tag{3.1}$$

where $Vol(P_i)$ is the volume of the Voronoi cell of P_i in the Voronoi diagram of all the control points and $Vol'_p(P_i)$ and $Vol'_p(p)$ are those in the Voronoi diagram of Q'_p. Note that $\sum_{k=1}^{N_p} u_k = 1$ and $u_k > 0, \forall k, 1 \leq k \leq N_p$. These coordinates are "natural" in the

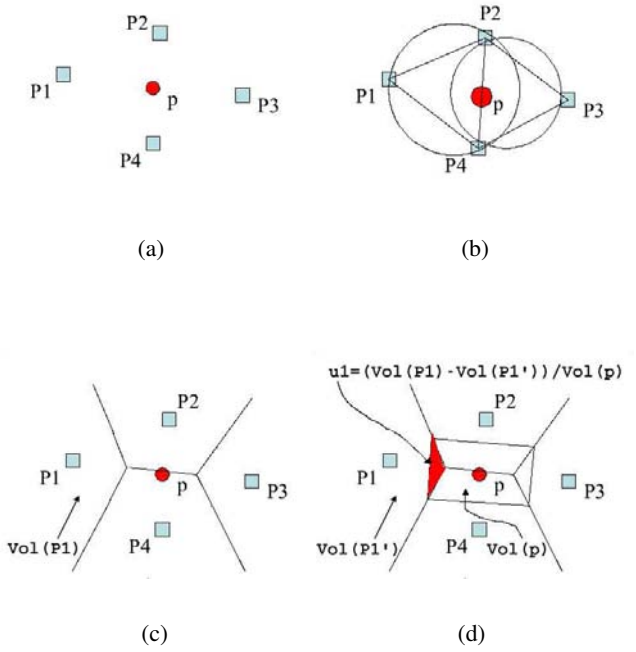


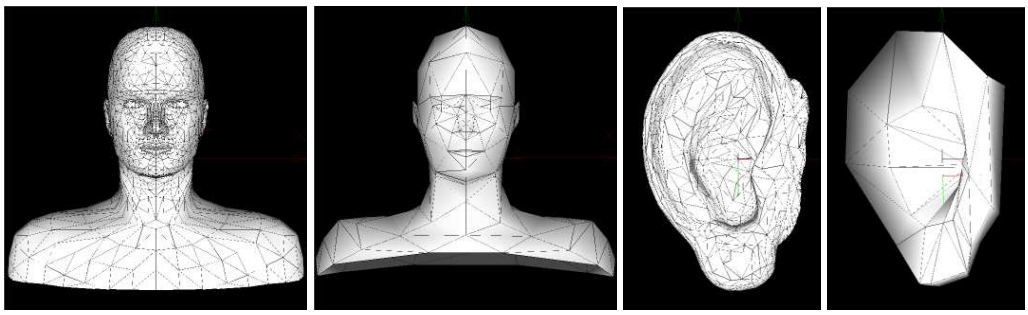
Figure 3.1: Sibson coordinates. (a) Subset of control points $Q_p = \{P_1, P_2, P_3, P_4\}$ surrounding mesh vertex p. (b) Delaunay triangulation of the control point set with circumscribed spheres around each Delaunay facet. (c) Corresponding Voronoi diagram. (d) Voronoi diagram for set $Q_p^{'} = \{p, P_1, P_2, P_3, P_4\}$. The Sibson coordinate for control point P_1 is proportional to the area shaded in gray.

sense that points in P that are closer to the point p have greater influence on it because the corresponding u_k is larger.

3.1.2 Introducing Deformations

As discussed above, our goal is to deform a *surface mesh* using the vertices of a much sparser *control mesh*, as our control points. We therefore take the control points to the vertices of the control triangulation complemented by the corners of the surface triangulations bounding box, so as to guarantee that the whole object is contained in their convex hull.

Let Q be this set of all control points and Q_S the set of vertices of the surface triangulation. For each point $p \in Q_S$, we find its natural neighbors $Q_p \subset P$ and the corresponding Sibson coordinates. This is referred to as freezing the control mesh to the object. Once computed, Sibson coordinates do not need to be changed when the object is deformed. When we move some of the control points from the set Q_p, the displacement of the model



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

Figure 3.2: Surface and control triangulations. (a,c) The generic low-resolution triangulation we use of upper-body and the ear modeling. (b,d) A subset of its vertices serve as DFFD control points, as discussed in this chapter. They are themselves triangulated to impose the regularization constraints of Chapter 5.

points is computed as follows:

$$\Delta p = \sum_{k=1}^{N_p} \Delta P_k u_k \tag{3.2}$$

where ΔP_i is displacements of control point from $P_k \in Q_p, k = 1, \ldots, N_p$. Finally, new object point position is computed as:

$$p^{new} = p + \Delta p, \tag{3.3}$$

In short, the deformations are local and defined by the natural neighbors, which helps to improve the flexibility of the approach and the realism in the final results.

In practice, we use a generic model of the human upper body, and its decimated version as the control mesh. In the Fig. 3.2 we show this meshes together with the example of the generic ear model and its accompanied control mesh.

3.2 Fitting DFFD Parameterized Models

In this section, we introduce the framework we have developed to fit *surface models* such as the ones of Fig. 3.2(a,c) to noisy image data. Our goal is to deform the surface—without changing its topology, that is the connectivity of its vertices—so that it conforms to the image data. In this work data is made of 3–D points computed using stereo. In standard least-squares fashion, for each stereo data point $\mathbf{x}_i$, we write an observation equation of the form $d(\mathbf{x}_i, \mathbf{S}) = y + \epsilon_i$, where $\mathbf{S}$ is a state vector that defines the shape of the surface, d is the distance measured from the data point $\mathbf{x}_i$ to the surface defined by state vector

$\mathbf{S}$, y is distance estimation we want to obtain, that is usually taken to be zero. ϵ_i is the deviation of the measured to the estimated distance value. In practice $d(\mathbf{x}, \mathbf{S})$ is taken to be the orthonormal distance of $\mathbf{x}$ to the closest surface triangulation facet. This results in $nobs$ such observations forming a vector

$$F(\mathbf{S}) = [..., d(\mathbf{x}_i, \mathbf{S}) - y, ...]^t_{1 \leq i \leq nobs} \tag{3.4}$$

that we minimize in the least squares sense by minimizing its square norm

$$\chi^2 = 1/2||F(\mathbf{S})||^2 \quad .$$

In theory we could take the parameter vector $\mathbf{S}$ to be the vector of all x, y, and z coordinates of the surface triangulation. However, because the image data is very noisy, we would have to impose a very strong regularization constraint. For example, we have tried to treat the surface triangulation as finite element mesh. Due to its great irregularity and its large number of vertices, we have found the fitting process to be very brittle and the smoothing coefficients difficult to adjust. This is why we chose to use the DFFD deformation approach instead.

3.2.1 DFFD parameterization

We therefore introduce *control triangulations* such as the ones of Fig. 3.2(b,d). Their vertices are points located at characteristic places on the human head or ear and defining their rough shapes and serve as DFFD control points. Some of these control points also are vertices of the surface model, while other are simply close to it and either inside or outside of it. This ability to place the control points is unique to DFFDs as compared to all other kinds of FFDs. The control triangulation facets will be used to introduce the regularization constraint discussed below. We tried to use several levels of resolutions of control meshes, but we found that increasing the number of control points does not influence final results of the deformation. This means that keeping low number of the control points, as we did, greatly saves time for computation.

In our scheme, we take the state vector $\mathbf{S}$ to be the vector of 3-D displacements of the DFFD control points, which is very natural using the DFFD formalism: As discussed in Section 3.1.2, we first freeze the control mesh to the model vertices. This means that for each vertex on the model we compute the influence of certain subset of control points. Those influences are expressed in terms of the Sibson coordinates from Section 3.1.2 and allows us to express displacement of every model vertex as the linear combination of displacements of control points which influence them.

3.2.2 3D observations

We use several sets of stereo pairs or triplets from the sequence of images of a given object as our input data such as those of Fig. 3.3(a,b,c). We then ran a simple correlation-based algorithm [45] to compute a disparity map for each pair or triplet and by turning each valid

disparity value into a 3–D point. This resulted in a large cloud of 3–D points that form an extremely noisy and irregular sampling of the underlying global 3–D surface. To reduce the size of the cloud and begin eliminating outliers, we robustly fitted local surface patches to the raw 3–D points [46]. We then fed the centers of those patches, shown in Fig. 3.3(d), as input to our surface fitting algorithm.

The center of each patch can then be treated as an attractor. The easiest way to handle this is to model it as a spring attached to the mesh vertex closest to it. This, however, is inadequate if one wishes to use facets that are large enough so that attracting the vertices, as opposed to the surface point closest to the attractor, would cause unwarranted deformations of the mesh. This is especially important when using a sparse set of attractors. In our implementation, this is achieved by writing the observation equation as:

$$d_i(\mathbf{x}_i, \mathbf{S}) = y + \epsilon_i \tag{3.5}$$

where $d_i(\mathbf{x}_i, \mathbf{S})$ is the orthogonal distance of the attractor to the closest facet, whose nominal estimated value y is zero. It can be computed as a function of the *x*, *y*, and *z* coordinates of the vertices of the facet closest to the attractor.

Because some of the observations, derived on the way explained above, may be spurious, we weigh them to eliminate outliers. Weighting is done as the preprocessing step, before the real fitting is started. In each iteration after fitting is done, we recompute the attachments and also recompute the observation weight w_i and take it to be inversely proportional to the initial distance $d_i(\mathbf{x}_i, \mathbf{S})$ of the data point to the surface triangulation. More specifically we compute w_i weight of the $\mathbf{x}_i$ as:

$$w_i = exp(\frac{d_i}{\overline{d_i}}), 1 \leq i \leq n \tag{3.6}$$

where $\overline{d_i}$ is the median value of the d_i. In effect, we use $\overline{d_i}$ as an estimate of the noise variance and we discount the influence of points that are more than a few standard deviations away.

3.2.3 2D observations

In our optimization framework besides 3D stereo observations we introduce also 2D observations. For each vertex (x_i, y_i, z_i) of the *surface triangulation* whose 2D projection in image j is (u_i^j, v_i^j) is known, we can write two observation equations:

$$P_{ru}(x_i, y_i, z_i) = u_i^j + \epsilon_i^u$$

$$P_{rv}(x_i, y_i, z_i) = v_i^j + \epsilon_i^u$$

where P_{ru} and P_{rv} stand for the projection in *u* and *v*. In this way we do not need the explicit 3D position of these feature points, only their 2D image location.

3.2.4 Regularization

Because there are both noise and potential gaps in the image data, we found it necessary to introduce a regularization term comparable the one proposed in [4]. Since we start with a generic model, we expect the deformation between the deformed shape and the initial one to be smooth. This can be effectively enforced by preventing deformations at neighboring vertices of the control mesh to be too different. If the control points formed a continuous surface parametrized in terms of two parameters u and v, a natural choice would therefore be to take this term to be

$$\begin{aligned} \mathcal{E}_D &= \sum_{s\in x,y,z} \mathcal{E}_{Ds} \\ \mathcal{E}_{Ds} &= \iint \left(\frac{\partial}{\partial u}\delta_s(u,v)\right)^2 + \left(\frac{\partial}{\partial v}\delta s(u,v)\right)^2 du\, dv \ , \end{aligned} \tag{3.7}$$

where $\delta_s(u,v)$ stands for the displacement along the x,y or z coordinates at each point of this control surface. In fact, the control surface is a triangulated one and we only have deformation values at the vertices. We are therefore use a finite element approach to compute $\mathcal{E}_{Ds}$as shown in the following.

3.2.4.1 Stiffness Matrix for C^0 Triangular Elements

We write the $\mathcal{E}_{Ds}$ term of Eq. 3.7 as

$$\mathcal{E}_{Ds} = \lambda/2 \sum_{1\le j\le n} \mathcal{E}_{D_s^j}$$

where $\mathcal{E}_{D_s^j}$ represents a summation over a facet j and λ is a regularization coefficient. In fact, we only know the deformations s_1^j,s_2^j and s_3^j at the vertices of the facet and we treat it as a C^0 triangular element. Over this specific facet, we write

$$\delta_s^j(u,v) = (1-u-v)s_1^j + us_2^j + vs_3^j \tag{3.8}$$

where $u,v \in [0,1]$ and $u+v<1$. It is then easy to show that $\mathcal{E}_{D_s^j}$ is the quadratic term

$$[s_1 s_2 s_3]\, K_s^j \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \ ,$$

where K_s^j is a 3×3 symmetric matrix that only depends on the shape of the triangle and, therefore, does not change during the optimization. These matrices can be summed into a global *stiffness matrix* K_s so that $\mathcal{E}_{Ds}$ becomes

$$s^t K_s s$$

where s stands for the vector of displacements at each vertex in one of the three coordinates. By summing these three terms, we obtain the final quadratic form or our complete regularization term

$$\mathcal{E}_D(\mathbf{S}) = \frac{\lambda}{2}\mathbf{S}^t K \mathbf{S} \tag{3.9}$$

where S is the complete state vector.

3.2.4.2 Incorporating the Stiffness Matrix into the Least-Squares Framework

We use the Levenberg-Marquardt algorithm [93] to iteratively minimize the square-norm of the observation vector $F(\mathbf{S})$ of Eq. 3.4. At each iteration, given the current state $\mathbf{S}$, the algorithm attempts to find a step $\mathbf{dS}$ that minimizes

$$\chi^2(\mathbf{S}+\mathbf{dS}) = 1/2||F(\mathbf{S}+\mathbf{dS})||^2 = 1/2F(\mathbf{S}+\mathbf{dS})^t F(\mathbf{S}+\mathbf{dS}) \ . \tag{3.10}$$

At the minimum, we should have

$$\begin{aligned} 0 &= \frac{\partial \chi^2}{\partial \mathbf{dS}} \\ &= A^t F(\mathbf{S}+\mathbf{dS}) \\ &\approx A^t(F(\mathbf{S}) + A\mathbf{dS}) \ , \end{aligned}$$

where A is the Jacobian of F. $\mathbf{dS}$ is therefore taken to be the solution of

$$A^t A \mathbf{dS} = -AF(\mathbf{S}) \ . \tag{3.11}$$

Adding a regularization term means that instead of minimizing simply the χ^2 term of Eq. 5.2, we minimize

$$\chi^2(\mathbf{S}) + \mathcal{E}_D(\mathbf{S}) = \frac{1}{2}||F(\mathbf{S}+\mathbf{dS})||^2 + \frac{\lambda}{2}(\mathbf{S}+\mathbf{dS})^t K(\mathbf{S}+\mathbf{dS}) \ . \tag{3.12}$$

At each iteration, we therefore solve

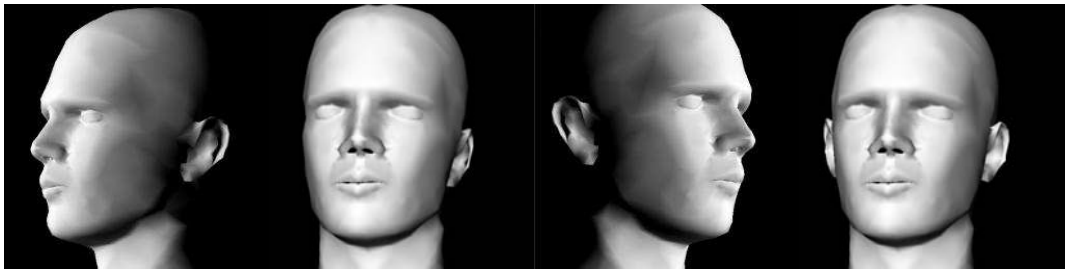
$$\begin{aligned} 0 &= \frac{\partial \chi^2}{\partial \mathbf{dS}} + \lambda K(\mathbf{S}+\mathbf{dS}) \\ &\approx A^t(F(\mathbf{S}) + A\mathbf{dS}) + \lambda K(\mathbf{S}+\mathbf{dS}) \ . \end{aligned}$$

$\mathbf{dS}$ therefore becomes the solution of

$$(A^t A + \lambda K)\mathbf{dS} = -A^t F(\mathbf{S}) - \lambda K \mathbf{S} \ . \tag{3.13}$$

Note that solving Eq. 3.11 or 3.13 involves the same amount of computation so that our regularization scheme adds very little to the computational complexity of the algorithm. Note also that the proposed optimization scheme is a semi-implicit one very similar to the one proposed for the original active contours [70] and that greatly improves the convergence properties of the algorithm.

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



(e) (f) (g) (h)

Figure 3.3: Calibrated video sequence: (a,b,c) Three images chosen from the calibrated video sequence, courtesy of IGP, ETH Zürich. (d) Centers of local surface patches fitted to the raw stereo data. (e,f,g) Automatically obtained shaded model after fitting to the stereo data and projected using the same perspective transform as that of the images. (h) Shaded model after interactive correction of the ears.

3.3 Results

We demonstrate and evaluate our technique mainly in the context of complete head, that is including face, ears and neck, from calibrated and uncalibrated video sequences.

3.3.1 Calibrated Video Sequence

We first illustrate the effectiveness of our approach using relatively clean stereo data. We use the sequence of forty 512x512 images where some are depicted by the first row of Fig. 3.3. They were acquired with a video camera over a period of a few seconds by turning around the subject who was trying to stand still. Camera models were later computed using standard photogrammetric techniques at the Institute for Geodesy and Photogrammetry,

ETH-Zürich. The centers of the local surface patches fitted to the raw stereo data shown on Fig. 3.3(d) are used as an input to our surface fitting algorithm.

We initialized the model by manually picking the five 2–D points overlaid in Fig. 3.3(b). We used them to compute a 4x4 rotation-translation matrix Rt such that five specific 3–D points on the generic model—outside corners of the eyes, corners of the mouth and tip of the nose—once multiplied by this matrix project as close as possible to the hand-picked 2–D location. Because these points are not coplanar, this guarantees that, when we multiply the generic model by this Rt matrix, we obtain an initial head model that is roughly aligned with the point cloud. We use it as the surface model that we deform using our DFFD approach, yielding the results shown in Fig. 3.3(e,f,g). In this case we did not use additional 2D observations provided manually. The resulting shape corresponds to the real head shape except around the ears that stick out more from the real head then from the reconstructed model. The reason for this behavior is because of well known fact that minimizing orthonormal distance of data points to the closest surface triangulation facet may have difficulty in deforming the model into concave objects [126]. More accurate deformation can be obtained when 2D projected observation are included in the objective function what is used in the examples of uncalibrated video sequence Fig. 3.4. One of the advantages of DFFD is that this can be fixed manually very quickly and very intuitively by moving one control point per ear to produce the model of Fig. 3.3(h).

3.3.2 Uncalibrated Video Sequence

Fig. 3.4 depicts two examples of reconstruction from uncalibrated images. First row shows images from different image sequences: one image from the stereo pair of images Fig. 3.4(a) and three frames from other uncalibrated video sequence. In both cases, we had no calibration information about the camera or its motion. We therefore used a model-driven bundle-adjustment technique [47] to compute the relative motion and, thus, register the images. We then used the same technique as before [46] to derive the clouds of 3–D points depicted by Fig. 3.4(e,f,g,h). Because we used fewer images and an automated self-calibration procedure as opposed to a sophisticated manual one, the resulting cloud of 3–D points is much noisier and harder to fit. Shaded models obtained after the fitting are depicted on Fig. 3.4(i,j,k,l). Notice they shaded models are shown in the same projection as the corresponding images on Fig. 3.4(a,b,c,d). In Fig. 3.4(d) and (l) we overlay on both the original image and the shaded projection of the mask the outlines of the face. Note that they correspond to the outlines predicted from the recovered 3–D geometry, thus indicating a good fit. These models can also be reused to resynthesizes textured images such as the ones of Fig. 3.5. We also animated generated models after the automatic fitting procedure using DFFD and produced complex facial expressions Fig. 3.5(i,j).

Least-square adjustment is applied in several iterations for the same value of regularization parameter λ. However, we tested how fitting to the uncalibrated data Fig. 3.4(b,c,d) is influenced by different choice of regularization parameter, in our case ranging from $\lambda = 1.0$ to $\lambda = 10$, and checked median error of the model to observations distance for certain number of iterations what is depicted on the Fig. 3.6(a). It is easy to see that final results do not

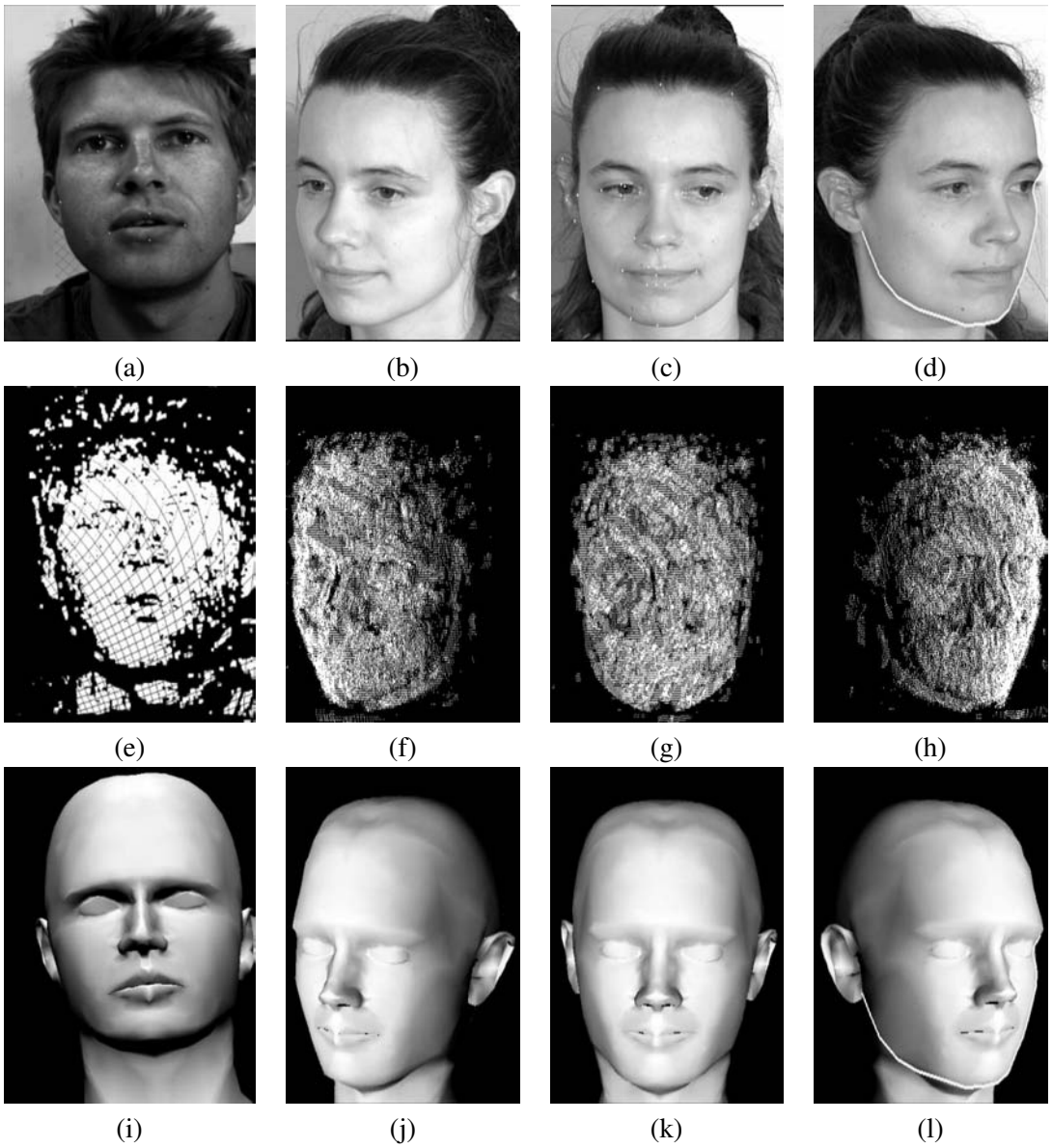
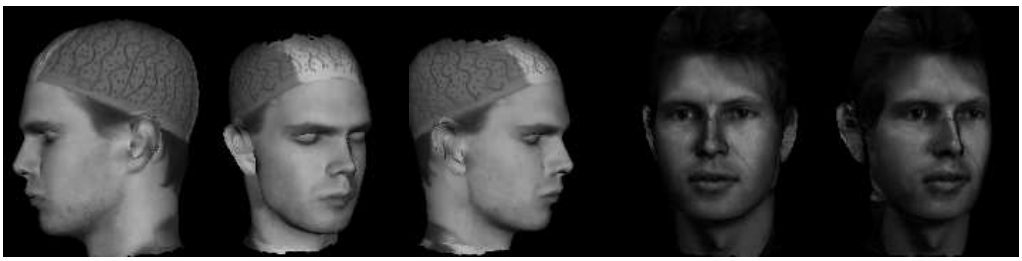


Figure 3.4: Uncalibrated video sequence: (a) One image from the stereo pair of images with overlayed manually provided 2D observations. (b,c,d) Three images from the uncalibrated video sequence out of eight images. On the image (c) 2D observations are depicted. (d) The image with overlaid face outline. (e,f,g,h) Centers of local surface patches fitted to the raw stereo data. (i,j,k,l) Automatically recovered shaded head model projected using the (a),(b),(c) and (d) image camera models. (l) Face outlines overlaid on the shaded projection in image (d).



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

(f) (g) (h) (i) (j)

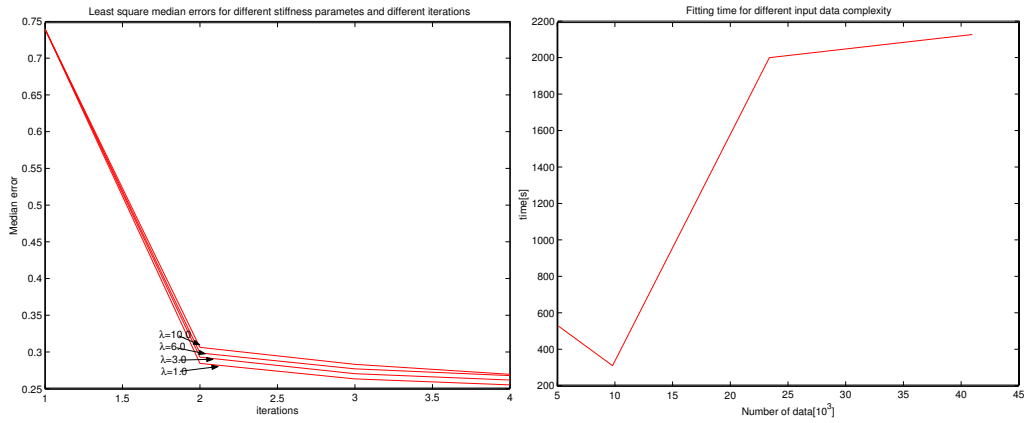
Figure 3.5: Textured models we created from the calibrated video sequence: (a,b,c), from stereo pair: (d,e) and from uncalibrated video sequence: (f,g,h); Animated model showing complex facial expressions (i,j)

depend on the choice of the regularization parameter since the median error of the model to observations distance does not greatly change with the increasing of the regularization parameter.

3.3.3 Performance measures

In our framework we use generic model of the human head including ears and neck identical to head-neck part of Fig. 3.2(a), which consists of 1396 vertices and 2756 facets. This generic model is used in all tests we performed to model heads. Control mesh is the one identical to the head-neck part from Fig. 3.2(b). System is tested on ancient Silicon Graphic Octane work station with R12000 processor working on 300MHz, and with 512Mb RAM memory.

The process starts with freezing the control mesh to the surface triangulation computing necessary Sibson coordinates. This is done once at the very beginning and it is used for all input data. Freezing the control mesh of the head takes 65s. On the Fig. 3.6(b) is shown



(a) (b)

Figure 3.6: (a) Least square median error for different regularization parameter λ ranging from *1.0* to *10.0* in respect to the number of iterations for video sequence whose three images are Fig. 3.4(b,c,d). (b) Fitting time in respect to the input data complexity

how the time for fitting depends on complexity of input data. Fitting time increases with the complexity of the data, but also depends on its configuration. For this test the number of fitting steps is fixed to three iterations and the stiffness parameter is set to $\lambda = 1.0$. Input data of the size 10^3, are fitted for the shorter time then the one of the lower complexity $5 \cdot 10^3$, since its configuration is initially closer to the generic model, so the least-square minimization converges faster.

3.4 Summary

In this chapter, we proposed to use the powerful DFFD extension to the conventional FFD shape deformation approach to fit deformable surface models to noisy 3–D image data. DFFDs give us the ability to place control points at arbitrary locations rather than on a regular lattice, and thus much greater flexibility.We demonstrated the effectiveness and robustness of our technique in the context of complete head modeling. We also showed that, in fact, we can model any complex shape for which a deformable triangulated model exists. For the specific application we chose, DFFDs offer the added benefit that they can be used to animate the head models we create and to produce realistic human expressions. The proposed framework is later extended to upper body motion tracking. In the following chapters we address the possibility of using DFFDs to deform implicit surfaces as opposed to the explicit ones we use here. In unrelated body-modeling work, we found that implicit

surface formulations lend themselves extremely well to fitting to the kind of data because they allow us to define a distance function of data points to models that is both differentiable and computable without search. Combining both approaches should therefore produce an even more powerful modeling tool.

4 Implicit Meshes

In the world of Computer Graphics, 3–D objects tend to be modeled as explicit surfaces such as polygonal meshes, parametric surfaces or subdivision surfaces. Because such representations are intuitive and easy to manipulate, they are widely accepted among graphics designers. These representations, however, are not necessarily ideal for fitting surfaces to potentially noisy and incomplete data such as 3–D points produced by laser-scanners and stereo systems or 2–D points from image contours. Fitting typically involves finding the facets that are closest to the 3–D data points or most likely to be silhouette facets, which introduces non-differentiabilities that degrade the convergence properties of most optimizers.

Implicit surfaces have received substantial attention in both the Computer Graphics and Computer Vision communities. They are well-suited for simulating physically based processes [115, 117, 83, 88] and for modeling smooth objects [2, 118, 40]. Because the algebraic distance to an implicit surface is differentiable, they do not suffer from the drawbacks discussed above when it comes to fitting them to 2 and 3–D data [110, 90, 31]. However, they have not gained wide acceptance, in part because they are more difficult to deform and to render than explicit surfaces.

In short, explicit surface representations are well suited for graphics purposes, but less so for fitting and automated modeling. The reverse can be said of implicit surface representations. In this thesis, we propose to combine the strengths of both approaches and to avoid their drawbacks by:

1. transforming explicit surfaces into implicit surfaces, whose shape closely approximates that of the original triangulations

2. deforming the implicit and the explicit surfaces in tandem for fitting and rendering purposes

This tandem of explicit and implicit surface we call *implicit mesh*. This is similar to the distance and convolution surfaces of [11], but makes up their major disadvantage, that is nonexistence of unique closed form mathematical expression describing the surface. Having such compact representation of an arbitrary surface and controlling its shape with certain relatively small number of external parameters, opens a number of possibilities of using different parameterizations for efficient automated fitting thanks to good mathematical properties of implicit surfaces.

To create the implicit surface we attach spherical or triangular volumetric primitives to each facet of the explicit mesh as shown in Fig 4.1. The parameters of those metaballs are a function of the facet geometry. As a result, when a facet deforms, so does the corresponding

metaball. In this chapter, we use the deformation technique, called Dirichlet Free Form Deformation (DFFD) [84, 59] introduced in previous Chapter 3, to control the shape, but in general, since we can turn any mesh into its implicit representation one could have chosen other methods, such as Free Form Deformations(*FFD*s) [100, 22], B-splines, subdivision or PCA parameterization [7] to deform the explicit mesh and consequently the implicit one.

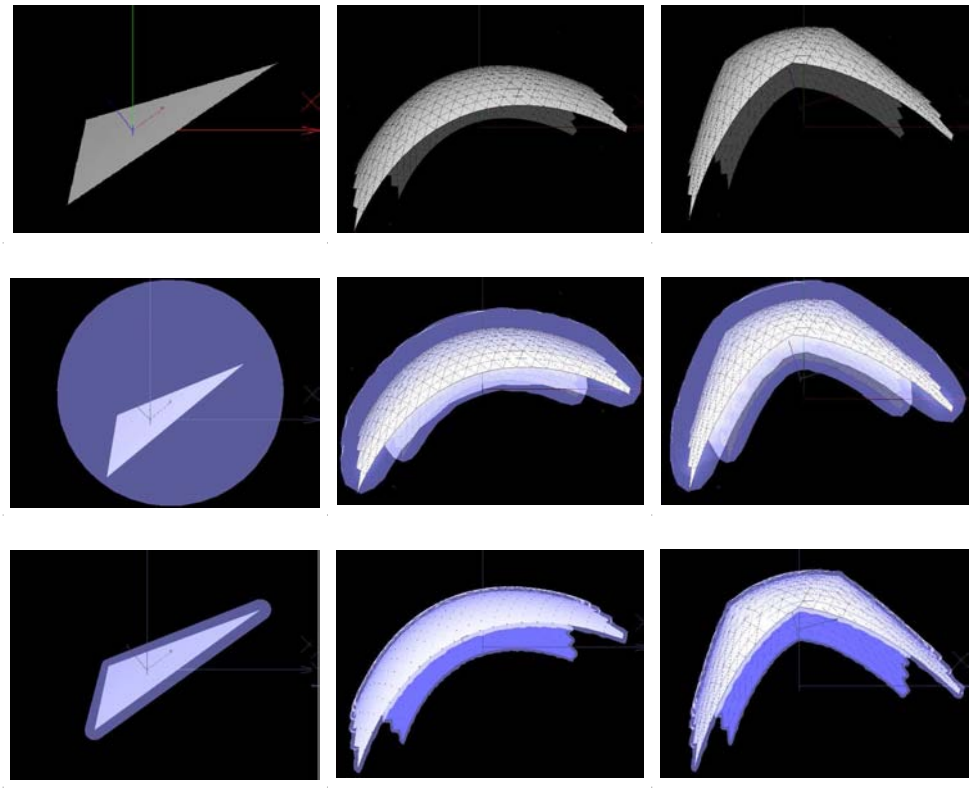


Figure 4.1: Converting an explicit surface into an implicit surface. Top row: From left to right: initial explicit mesh facet, triangulated mesh, and deformed mesh. Middle row: Explicit mesh converted into an implicit one using spherical primitives. Bottom row: Explicit mesh converted using triangular primitives.

Our contribution is therefore an approach to surface reconstruction that allows to take an arbitrary explicit surface model of any complexity, for example one that has been obtained from the web and was not designed with fitting in mind, turn it into an implicit surface, and deform it to obtain an optimal fit to image-data. Because the implicit surface closely approximates the explicit one, we can keep the deformed explicit mesh and use it instantly for rendering.

In the remainder of the chapter, we introduce our approach to creating implicit meshes and deforming them. In more details we describe construction of spherical and triangular implicit meshes, their smoothing and shape control.

4.1 Implicit Mesh Models

To create an implicit mesh that can deform in tandem with an explicit one, we define an implicit surface that closely approximates the explicit shape and whose deformations depend only on the motion of the explicit mesh vertices.

To this end, we attach a volumetric primitive, or metaball, to each facet. This can be done in two different ways. The simplest is to use spherical primitives, such as those depicted by the middle row of Fig. 4.1, which are only adapted to fairly regular and high resolution meshes. A more sophisticated approach requires using the triangular metaballs depicted by the bottom row of Fig. 4.1, which are more complex but can be used to accurately approximate arbitrarily low-resolution or irregular meshes. We describe these two kinds of metaballs, and implicit meshes created from them, in more detail below.

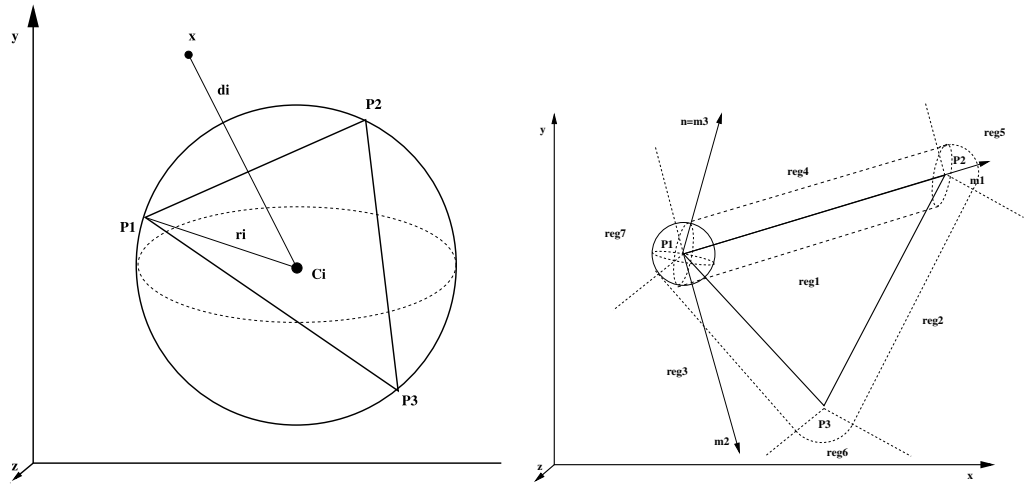


Figure 4.2: Triangular facet enclosed with the (a) Spherical metaball and (b) Triangular metaball, labeled according to the Eqs. 4.1 and 4.6.

4.1.1 Spherical Implicit Meshes

A spherical metaball [61] is created by circumscribing a spherical primitive around a facet in such a way that the sphere center $\mathbf{C_i}$ lies on the facet and corresponds to the center of the circumscribed circle around the facet, as shown in Fig 4.2(a). For facet Fi, it defines a potential field that can be expressed as:

$$f_i(\mathbf{x}) = exp(-k(d_i(\mathbf{x}) - r_i)) \ , \tag{4.1}$$

where $\mathbf{x}$ is a 3–D point, $d_i(\mathbf{x})$ is the Euclidean distance of $\mathbf{x}$ to $\mathbf{C_i}$, r_i is the radius of the spherical metaball, and k is a parameter that controls the smoothness of the overall implicit

surface and will be discussed below. C_i is determined from:

$$\mathbf{C_i} = \frac{3\mathbf{G_i}(P_1, P_2, P_3) - \mathbf{H_i}(P_1, P_2, P_3)}{2} \quad , \tag{4.2}$$

where $\mathbf{G_i}$ is facet's center of gravity, and $\mathbf{H_i}$ its orthocenter, both defined by the facet's vertices P_1, P_2 and P_3. We can then write

$$d_i(\mathbf{x}) = \|(\mathbf{x} - \mathbf{C_i})\| \quad , \tag{4.3}$$

$$r_i = \|(\mathbf{P_1} - \mathbf{C_i})\| \quad . \tag{4.4}$$

Note that both $\mathbf{C_i}$ and r_i depend only on the positions of the facet vertices.

The implicit mesh is taken to be an isosurface of the sum of all these potential fields. Formally, this isosurface is the set of 3–D points $\mathbf{x} \in R^3$ that satisfy

$$F(\mathbf{x}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}) - r_i)) = 0 \quad , \tag{4.5}$$

where T is an arbitrarily chosen isovalue. Usually we take T to be one, so that all points on the surface have a zero potential field value, the values smaller than zero are inside and those greater than zero outside. Given the fact that each separate exponential field function decreases quickly, when the model is close enough to the data, we can speed up the computation by introducing an adaptive threshold t on the distance above which the function does not need to be evaluated as depicted in Fig. 4.3.

The isosurface is approximately parallel to the mesh on both sides and encloses the volume shown in gray in the middle row of Fig. 4.1. Its thickness is a function of the r_i radii of the metaballs attached to the individual facets, and therefore of their sizes. As shown in Fig. 4.3, for values of k greater than one, the exponential drops fast and the individual metaballs have influence only over a relatively short range. As a result, the isosurface closely follows the shape of the metaballs and can be very bumpy if the facets are irregular. For values of k smaller than one, the metaballs tend to blend into each other, which yields a smoother but thicker isosurface. The first is desirable but the second can result in the problems depicted by Fig. 4.5. In practice, we found that by setting k to values around 1.0 both requirements are fulfilled.

Because the spherical metaballs are circumscribed around the facets their radius r_i depends on the size of the triangle. As shown in the second row of Fig. 4.1, as long as the explicit mesh is relatively regular or high resolution, this yields a valid approximation. However, because large facets produce large primitives, the approximation becomes much less accurate as the facets of the explicit mesh increase in size. When dealing with low resolution irregular meshes, such as the one of Fig. 4.4(a), elongated facets produce an implicit surface that enclose a volume whose thickness can change dramatically, as shown in Fig. 4.4(b).

Up to a point, that can be remedied by refining the mesh as shown in the bottom row of Fig. 4.4(a), so that it consists of many smaller size facets and produces the better approximation as depicted in Fig. 4.4(b). However, this results in a substantial computational

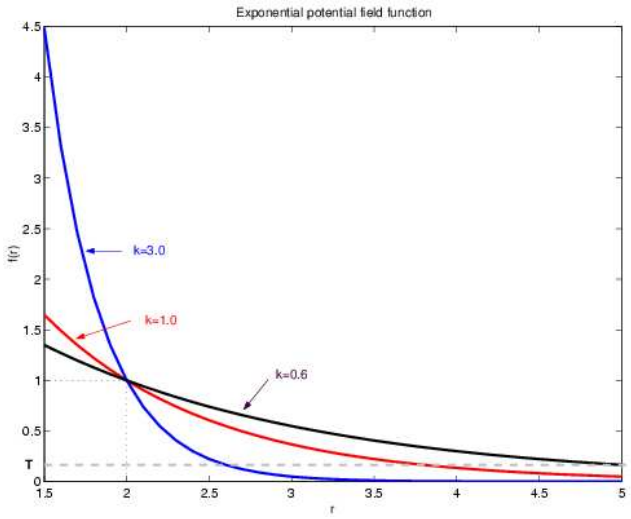


Figure 4.3: Exponential potential field function of one metaball, showing how the smoothing parameter k controls the range of influence of the primitive and, thus, the amount of smoothing.

cost increase. Furthermore, the volume enclosed by the isosurface remains relatively thick, unless the facets are made to be even smaller, which would become prohibitively expensive.

4.1.2 Triangular Implicit Meshes

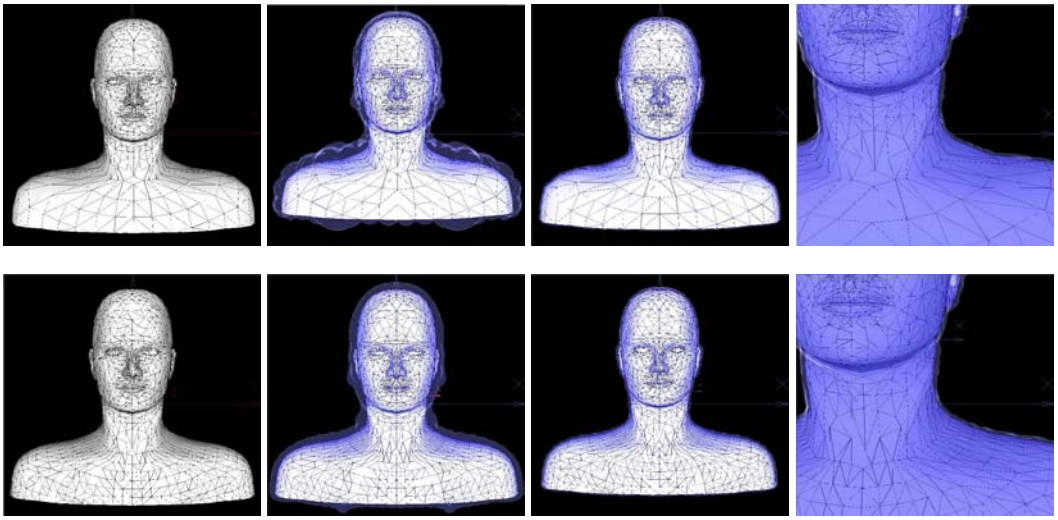
To solve these problems, and create implicit surfaces that more closely approximate arbitrary meshes, we can replace the spherical metaballs by *triangular* ones [61], such as those depicted by the bottom row of Fig. 4.1.

This is done by replacing the Euclidean distance to the facet's center of Eq. 4.3 by a distance $d(\mathbf{x})$ that more closely approximates the orthogonal distance to the whole facet. We could, of course, take $d(\mathbf{x})$ to simply be the orthogonal distance to the facet plane but that would mean that all points on that plane have a zero distance, no matter how far they are from the facet. Instead, we define $d(\mathbf{x})$ as a piecewise-polynomial function as follows.

For a given facet F_i, we compute the partition of the plane it defines in the seven regions depicted by Fig. 4.2(b). Given a point $\mathbf{x} \in R^3$, we compute its projection on the facet plane and, depending in which region it falls, we take its distance to F_i to be

$$d(x) = \begin{cases} \left(\frac{\mathbf{n}\bullet(x-P_1)}{\|\mathbf{n}\|}\right)^2 & , \quad \mathbf{x} \in reg1 \\ \frac{\|(\mathbf{x}-\mathbf{P}_i)\bullet\mathbf{e}\|^2}{\|\mathbf{e}\|^2} & , \quad \mathbf{x} \in reg2,\ reg3,\ reg4 \\ \|\mathbf{x}-\mathbf{P}_i\|^2 & , \quad \mathbf{x} \in reg5,\ reg6,\ reg7 \end{cases} \tag{4.6}$$

When the 3–D point $\mathbf{x}$ projects inside the facet, it falls within $reg1$ and $d(\mathbf{x})$ is simply the squared orthogonal distance to the facet plane, expressed in terms of its normal $\mathbf{n}$ and vertex



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

Figure 4.4: Conversion of low and high resolution explicit meshes to implicit ones. (a) Low and high resolution meshes. (b) Corresponding implicit surfaces created using spherical metaballs. The volume enclosed by the implicit surfaces is shown transparent. (c) Corresponding implicit surfaces created using triangular metaballs. The enclosed volume is still shown transparent but, for both the low and high resolution meshes, the implicit surface is now so close to the explicit one that it is almost impossible to see at this resolution. (d) Magnified view so that the small difference between the implicit and explicit meshes, which is a function of the d_0 parameter of Eq. 4.7, becomes visible.

P_1. If $\mathbf{x}$ projects outside of the facet but in the bands perpendicular to the edges, it falls within regions $reg2$, $reg3$ or $reg4$ and $d(\mathbf{x})$ becomes the squared Euclidean distance from the closest edge passing respectively through $P_i \in \{P_2, P_1, P_1\}$ whose direction is given by vector $\mathbf{e} \in \{P_1P_2, P_1P_3, P_2P_3\}$ respectively. In the remaining cases, the projection falls within regions $reg5$, $reg6$ or $reg7$ and $d(\mathbf{x})$ is taken to be the Euclidean distance to the closest vertex $P_i \in \{P_2, P_3, P_1\}$.

Fig. 4.6(a) depicts $d(x)$ for a standardized facet lying in the $z = 0$ plane with vertices $P_1 = \{0, 0, 0\}$, $P_2 = \{1, 0, 0\}$ and $P_3 = \{0, 1, 0\}$. Note that on the surface of the triangle distance is uniformly zero while along the edges and at the vertices, we find well-behaved parabolas that blend smoothly.

Note that the distance of a point to a facet's edge that appears on the second line of Eq. 4.6 is the cylindrical distance to that edge. Similarly, the distance to a vertex that appears on the third line of Eq. 4.6 is the spherical distance to that vertex. Intuitively, a triangular metaball can be understood as being made of two planes, one on each side of the explicit facet, that

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

Figure 4.5: Relationship between the accuracy of the approximation and the quality of the fitting results. (a) An explicit mesh is approximated using spherical metaballs, which results in an implicit surface of a certain thickness. (b) If the original mesh intersects the data represented by small circles, different sides of the implicit surface may become attracted to the data, resulting in a poor fit. (c,d) Using triangular primitives yields a much thinner implicit surface and a much improved fit.

blend seamlessly with three implicit cylinders whose axes are aligned with the edges at three implicit spheres centered at the vertices. The cylinders and spheres are represented by dotted lines in Fig. 4.2(b). Latter we formalize this observation in terms of a matrix representation. In Appendix 9.1, we then use this representation to give a formal proof that the distance of Eq. 4.6 is C^1 with respect to the 3–D coordinates of both the vertices and the $\mathbf{x}$ data points.

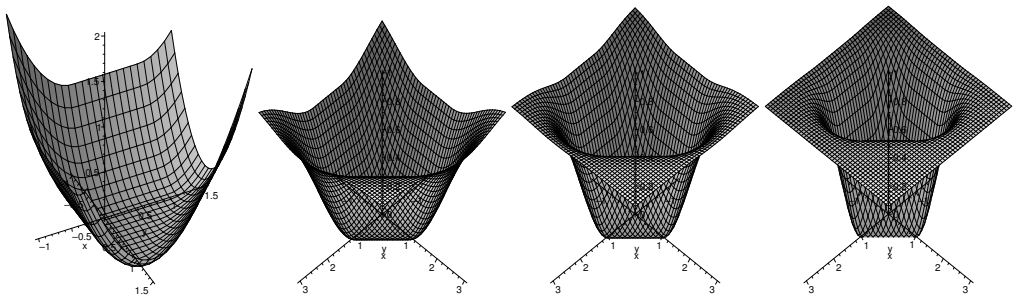
Finally, the distance function can be incorporated in the same potential field function as the one used for spherical metaballs

$$f_i(\mathbf{x}) = exp(-k(d_i(\mathbf{x}) - d_0^2)) \ , \tag{4.7}$$

where d_0 represents the constant thickness of the implicit surface and replaces the variable spherical radii r_i of Eq. 4.1. Fig. 4.6(b,c,d) depicts the potential field function of Eq. 4.7 for different values of smoothing parameter $k \in \{0.5, 1.0, 2.0\}$. In this case, potential filed for the points projecting on the facet (*reg1* of Fig. 4.2(b)) has constant values greater then zero, that results in function parallel to the facet plane at the height $exp(kd_0^2)$. The overall shape of the function is similar to the one of the distance function, but has different scale over the z-axis. Again, the total field is the sum of the individual metaballs fields, which yields the final expression of the implicit surface as the set of points $\mathbf{x} \in R^3$ such that

$$F(\mathbf{x}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}) - d_0^2) \ . \tag{4.8}$$

Note that now, unlike in the case of spherical metaballs, the behavior of the potential field has become independent from facet sizes and mesh resolution. It only depends on the C^1 distance function $d(x)$ and the d_0 and k parameters of Eq. 4.7 over which we have full control.



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

Figure 4.6: (a) Distance function of Eq. 4.6 and (b) potential field function of Eq. 4.7 with different values of parameters $k = \{0.5, 1.0, 2.0\}$, for one facet laing in xy-plane and fixed z coordinate.

As shown in Fig. 4.4, this lets us approximate arbitrary meshes much more closely, no matter how irregular they may be. As a result, the fitting failure mode depicted by Fig. 4.5(a,b) can now be overcome. As shown in Fig. 4.5(c, d), if d_0 is taken to be small enough, it does not really matter to which side of the implicit surface the data is closest because its thickness has become negligible.

4.1.3 Smoothing the Triangular Implicit Meshes

In practice, both the d_0 thickness and k smoothing parameters of Eq. 4.7 influence the smoothness and accuracy of the implicit mesh. To illustrate this, we computed zero level-sets around the three 2–D line segments aligned along the x-axis using a 2–D version of the distance function of Eq. 4.6. In Fig. 4.7(a), we plot the zero level-sets for fixed $d_0 = 0.1$ and varying k, and in Fig. 4.7(b) for fixed $k = 20.0$ and varying d_0. Increasing d_0 or decreasing k tends to smooth the implicit surfaces, but thickness of the volume they enclose becomes huge, which is highly undesirable as depicted in Fig. 4.7(a,b). Our goal is therefore to find the best possible compromise between accuracy and smoothness as a function of k and d_0. To quantify the influence of these two parameters, we introduce quantitative measures of smoothness and accuracy that both depend on the thickness of the volume enclosed by the implicit mesh.

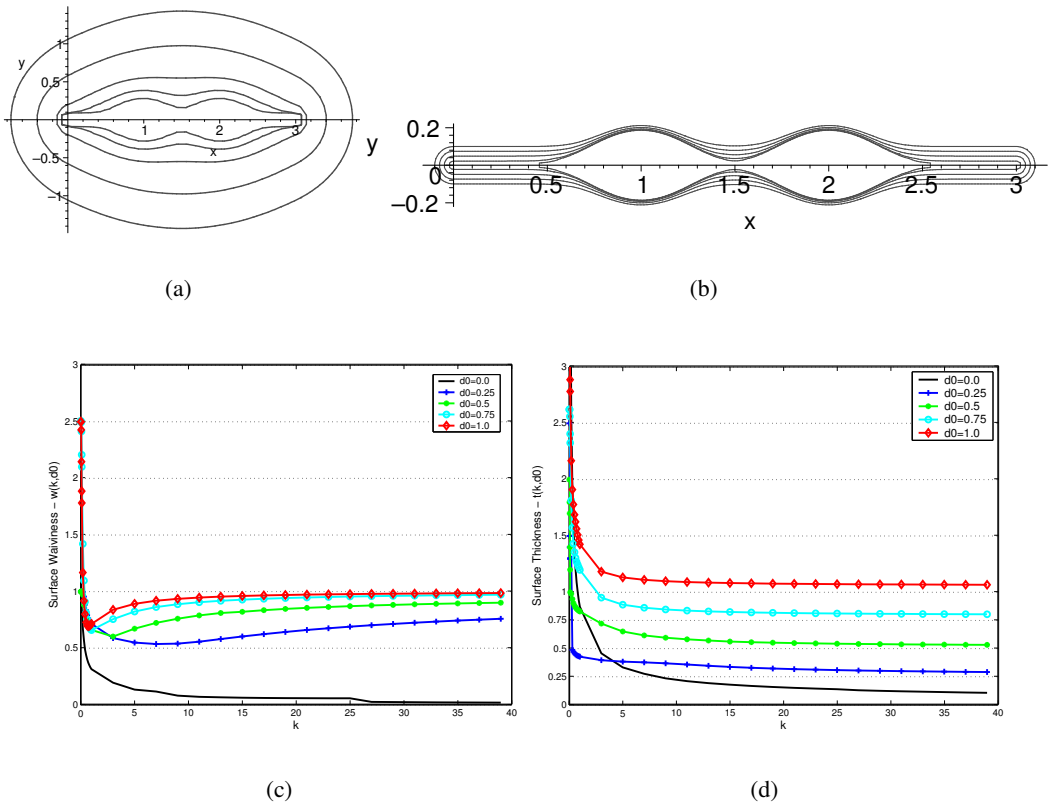


Figure 4.7: Top row: Zero level-sets of three unit line segments aligned along x-axis for (a) $d_0 = 0.1$ and varying k from 0.1 corresponding to the outer level-set, to 10.0 , corresponding to the inner level-set. (b) $k = 20.0$ and varying d_0 from 0.0 for the inner level-set to 0.1 for the outer level-set. Bottom row: Smoothness and accuracy as a function of the k parameter. (c) Surface waviness and (d) surface thickness as a function of k for different values of d_0.

- **Surface waviness**: Average ratio of minimal and maximal volume thickness evaluated respectively at the center of gravity of the facets and at their vertices:

$$w(k, d_0) = \frac{\frac{1}{Nf}\sum_{i=1}^{Nf} d_i^f}{\frac{1}{Nv}\sum_{i=1}^{Nv} d_i^v} = \frac{\frac{1}{Nf}\sum_{i=1}^{Nf} \|\mathbf{x}_i^{cn}(k, d_0) - \mathbf{x}_i^{cg}\|}{\frac{1}{Nv}\sum_{i=1}^{Nv} \|\mathbf{x}_i^{vn}(k, d_0) - \mathbf{x}_i^{v}\|}$$

- **Surface thickness**: Average volume thickness measured at the same locations as those used to evaluate waviness:

$$t(k, d_0) = \frac{1}{Nf}\sum_{i=1}^{Nf} \|\mathbf{x}_i^{cn}(k, d_0) - \mathbf{x}_i^{cg}\| + \frac{1}{Nv}\sum_{i=1}^{Nv} \|\mathbf{x}_i^{vn}(k, d_0) - \mathbf{x}_i^{v}\|$$

where $\mathbf{x}_i^{cn}$, $F(\mathbf{x}_i^{cn}) = 0$, is a point of the minimal volume thickness of the implicit mesh measured from the facet center of the gravity $\mathbf{x}_i^{cg}$, and $\mathbf{x}_i^{vn}$, $F(\mathbf{x}_i^{vn}) = 0$, is a point of the maximal volume thickness of the implicit mesh measured from the mesh vertex $\mathbf{x}_i^{v}$, and Nf, Nv being a number of facets and vertices respectively.

In Fig. 4.7(c,d), we plot these values as a function of k for different values of d_0 in the case of the explicit mesh of Fig. 3.2(b). The behavior if entirely consistent with the 2–D case depicted by the top row of Fig. 4.7. For each value of d_0 the waviness of Fig. 4.7(c) tends towards one when k is increased which means that the bulges disappear. However, the higher the value of d_0, the faster it goes to one. Similarly, as can be seen in Fig. 4.7(d), the thickness is large for small values of k and asymptotically approaches d_0 for huge ones. In practice, we constrain d_0 to be less then 10% of the average edge length and seek values of d_0 and k such that

$$w(k, d_0) > 1.0 - w_{max} \ , \tag{4.9}$$

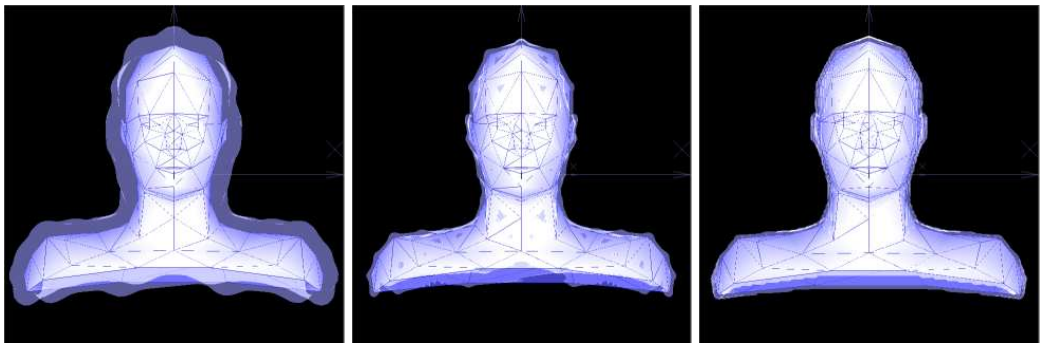
$$t(k, d_0) < d_0 + t_{max} \ , \tag{4.10}$$

where w_{max} and t_{max} are two user specified thresholds, and for which k appears to be minimal. This is important since choosing the high values for k will satisfy both conditions, however such huge values of k will quickly cut off the influence of the potential field and eliminate potential inliers. Note that for generic models such as the ones of Fig. 3.2(b), this computation needs only be performed once and the optimal values of k and d_0 stored and reused for all subsequent fits to image data.

Formally, the proposed algorithm can be formalized as follows:

Algorithm 1 Determining optimal value of k and thickness d_0 parameters

Require: $Mesh, max\_k, Step\_d_0, Step\_k, w_{max}, t_{max}$
Return: k^{min}, d_0^f
  $m\_res \leftarrow MeshRes(Mesh)$ {MeshRes returnes mesh's average edge length}
  $min \leftarrow +\inf$
  for $d_0 = 0.0$ to $0.1 * m\_res$ **step** $d0+ = Step\_d_0$ **do**
    for $k = 0.01$ to max_k **step** $k+ = Step\_k$ **do**
      if $w(k, d_0) > 1 - w_{max}$ **then**
        if $t(k, d_0) < d_0 + t_{max}$ **then**
          $K = k, D = d_0$
        end if
      end if
    end for
    if $K < min$ **then**
      $min \leftarrow K, k^{min} \leftarrow K, d_0^f \leftarrow D$
    end if
  end for



(a) (b) (c)

Figure 4.8: Implicit mesh for fixed $d_0 = 0.75$ and (a) $k = 0.01$, (b) $k = 5.0$, (c) $k = 10.0$ whose measured waviness and thickness values are shown in Fig. 4.7(c, d).

In order to demonstrate it on the real 3–D example of highly irregular mesh, we depicted how smoothness constant k influences quality of the approximation. In Fig. 4.8 we depicted implicit meshes for different values of the smoothness constant $k = \{0.01, 5.0, 10.0\}$ and fixed value of thickness constant $d_0 = 0.75$, that actually correspond to the quality measure graphics of Fig. 4.7(c, d). Our algorithm computed the best values for $k = 10$ and $d_0 = 0.75$.

4.1.4 Matrix Representation of the Triangular Metaballs

As discussed in Section 4.1.2 a triangular metaball, can be understood as being made of two planes, one on each side of the explicit facet, that blend seamlessly with three implicit cylinders whose axes are aligned with the edges and three implicit spheres centered at the vertices as shown in Fig. 4.2.

For a given facet F_i with vertices $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$, let us consider a local coordinate frame attached to it, with its origin at one of the vertices and x-axis aligned with one of the edges. It can be represented in matrix form as

$$M = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{P}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A & \mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix}, A = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \end{bmatrix}_{3x3}, \mathbf{c} = \mathbf{P}_1,$$

where $\mathbf{m}_1, \mathbf{m}_2$ and $\mathbf{m}_3$ are axes of the local coordinate frame, with $\mathbf{m}_1$ being aligned with the x-axis, and $\mathbf{m}_3$ being the facet's normal. A point $\mathbf{x}_l$ in the local coordinate frame is converted to point $\mathbf{x}_g = M\mathbf{x}_l$ in the global coordinate frame. Let $T_r = M^{-1} = \begin{bmatrix} A^T & -A\mathbf{c} \\ \mathbf{0} & 1 \end{bmatrix}$ be the inverse of M. So, a point $\mathbf{x}_g$ in the global coordinate frame is converted to point $\mathbf{x}_l = T_r\mathbf{x}_g$ in the local coordinate frame.

The coordinates of $\mathbf{P_1}$ are a function of the displacements of the control points and therefore of the state vector S of Eq. 4.12. Let $\mathbf{n}(S) = [n_1, n_2, n_3]$ be the facet normal, which also is a function of S, and let $\mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & x_3 & 1 \end{bmatrix}$ be the 3–D point in space for which the distance is computed. The distance function of Eq. 4.6 can now be expressed in matrix form as

$$d(\mathbf{x}, \mathbf{S}) = \begin{cases} d_p(\mathbf{x}, \mathbf{S}) = \mathbf{x}^T L_1 L_2 \mathbf{x} & , \quad \mathbf{x} \in reg1 \\ d_c(\mathbf{x}, \mathbf{S}) = \mathbf{x}^T T_r^T L_c T_r \mathbf{x} & , \quad \mathbf{x} \in reg2,\ reg3,\ reg4 \\ d_s(\mathbf{x}, \mathbf{S}) = \mathbf{x}^T T_r^T L_s T_r \mathbf{x} & , \quad \mathbf{x} \in reg5,\ reg6,\ reg7 \end{cases} \tag{4.11}$$

where

$$L_1(S) = \begin{bmatrix} n_1 & 0 & 0 & 0 \\ 0 & n_2 & 0 & 0 \\ 0 & 0 & n_3 & 0 \\ 0 & 0 & 0 & -\mathbf{n}^T \bullet \mathbf{P}_1 \end{bmatrix} \text{ and } L_2(S) = \begin{bmatrix} \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \\ \mathbf{n}^T & | -\mathbf{n} \bullet P_1 \end{bmatrix}$$

are matrices that define the plane. Similarly, the matrices

$$L_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } L_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

respectively define a cylinder aligned with the x-axis in the global coordinate frame and a sphere centered at the origin of the global coordinate frame.

Such matrix representation gives us an elegant and compact expression of the distance function. Its main advantage is easy derivative computation and its still compact representation. Although, matrix multiplication is expensive operation it is less complex to compute derivatives using matrix notation and for direct distance calculation to keep initial analytical expression of Eq. 4.6. The derivative computation will be addressed in the chapter concerning our optimization framework.

4.2 Deforming Implicit Meshes

The shape of the implicit meshes of Section 4.1 depends only on the position of the 3–D vertices of the corresponding explicit meshes. In theory, for fitting purposes, we could optimize the shape with respect to the x, y, and z coordinates of these vertices. However, in this case number of parameters can be very big and because image data is very noisy, we would end up with ill posed optimization problem. Instead, we show how we can use Dirichlet Free Form Deformations [84] to parameterize the surfaces, both explicit and implicit, in terms of the positions of a much smaller number of control points. The same can be said for any other potentially used parameterization.

In other words, for any given set of the shape parameters, the overall surface shape is entirely described by the state vector

$$\mathbf{S} = [a_1, \ldots, a_n] \tag{4.12}$$

formed by concatenating those parameters. In case of DFFD the state vector is composed of the displacements of all the control points with respect to their original positions. For PCA face models it is composed of PCA coefficients.

To deform an implicit surface created using either spherical or triangular metaballs, it is sufficient to change the parameters that define the shape of the primitives. Let us first consider one single facet and its attached spherical or triangular metaball. A spherical primitive is defined by the distance function of Eq. 4.1 and the r_i radius, which is a function of the vertex positions. Similarly, a triangular metaball is defined by the distance function of Eq. 4.6 and the d_0 thickness parameter. In both cases, because the positions of the vertices can be expressed as the weighted linear combination of Eq. 3.3 of the control points of displacements, or PCA coefficient, the distance functions $d(\mathbf{x})$ of Eqs. 4.1 and 4.6 depend not only on $\mathbf{x}$, the 3–D coordinates of the point whose distance is evaluated, but also on the control points. We therefore rewrite the distance function of Eq. 4.6 and Eq. 4.3 so to depend on state vector parameters of Eq. 4.12 like:

$$d(\mathbf{x}) = d(\mathbf{x}, \mathbf{S}) \quad , \tag{4.13}$$

where $\mathbf{S}$ is the state vector of Eq. 4.12.

As a consequence, when considering all the facets together, we can also rewrite the field potential functions F of Eqs. 4.5 and 4.8 that define the implicit mesh. When using triangular metaballs, F becomes

$$F(\mathbf{x}, \mathbf{S}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}, \mathbf{S}) - d_0^2)) \quad , \tag{4.14}$$

where $\mathbf{x}$ is a point in R^3 and d_i is the distance to facet i defined by Eq. 4.13. Similarly, for spherical metaballs, we write

$$F(\mathbf{x}, \mathbf{S}) = T - \sum_{i=1}^{N} exp(-k(d_i(\mathbf{x}, \mathbf{S})) - r_i(\mathbf{S}))) \quad , \tag{4.15}$$

since r_i also is a function of the vertex positions.

In this fashion, we have parametrized both the explicit and the implicit surface in terms of the $\mathbf{S}$ state vector. As discussed in the following section, this will allow us to deform both representations in tandem to fit the corresponding surface to image data by minimizing a differentiable objective function.

4.3 Summary

We have presented an approach to combining explicit and implicit surface representations that allows us to take advantage of the strengths of both. To this end, we have developed a technique for creating *implicit meshes* from explicit ones by attaching triangular or spherical primitives to their facets. These primitives are defined in such a way that their shape depends only on the 3D location of the mesh vertices, which will allow us to simultaneously fit both representations to image data by minimizing a differentiable objective function. We also showed how this representation can be efficiently controlled by any set of parameters, which control the underlying explicit mesh, for example by a set of DFFD control points of Chapter 3. The compact matrix formulation of triangular implicit mesh is given, announcing that it can be used for elegant derivative computation.

5 Optimization Framework

In this chapter, we introduce a framework we have developed to fit *generic models* such as ones of Fig. 3.2(a,c) to noisy data. Our goal is to deform such surfaces—without changing the connectivity of their vertices—so that it conforms to the image data, which here comes both, in the form of 3–D point clouds from stereo and 2–D silhouette points from occluding contours. Latter, in the following Chapter 6, we introduce the use of interest points in our optimization framework. To highlight the capabilities of our implicit meshes, we will perform the fitting using either the implicit or explicit representations in order to compare the results. This framework is used both for 3–D reconstruction and tracking of 3–D nonrigid motion of deformable objects. In case of our implicit meshes, using silhouettes is much more efficient than the classical way of using silhouettes with explicit polygonal meshes. The generic models we use can be parameterized differently. In this chapter we express their deformations in terms of the state vector $\mathbf{S}$ of Eq. 4.12. In practice this state vector contains DFFD control points, PCA coefficients or directly vertex coordinates of the model.

We start with a brief description of stereo and silhouette data retrieval. After, we address objective functions used for fitting implicit meshes and explicit surfaces to those data. Weighting data observations and their attachment to the generic model are further discussed, and a regularization of our optimization procedure is explained in details. In the end derivatives computation will be also addressed, where we especially want to demonstrate valuableness of the matrix notation used to express triangular implicit mesh distance function.

5.1 Stereo

Fig. 5.1(a,b,c) depicts three frames from uncalibrated video sequence. In this cases, as in all following once, we had no calibration information about the camera or its motion. We therefore used a model-driven bundle-adjustment technique [47] to compute the relative motion and, thus, register the images. The generic models of Fig. 3.2 are used in the bundle-adjustment procedure. They are initialized on one image where the subject is facing the camera. It is done according to, at least five, manually provided feature points that can be some of those provided in Fig. 5.1(b). We then used a simple correlation-based technique of [46] to derive the clouds of 3–D points depicted by Fig. 5.1(d,e,f). Those data points are centers of the local surface patches fitted to the raw stereo data and they are used, in part, as an input to our surface fitting algorithm. Because we used fewer images and an automated self-calibration procedure as opposed to a sophisticated manual one, the resulting cloud of 3–D points is much noisier and harder to fit. In some of our examples we used a more

sophisticated max-flow stereo algorithms [96] to produce high quality stereo data.

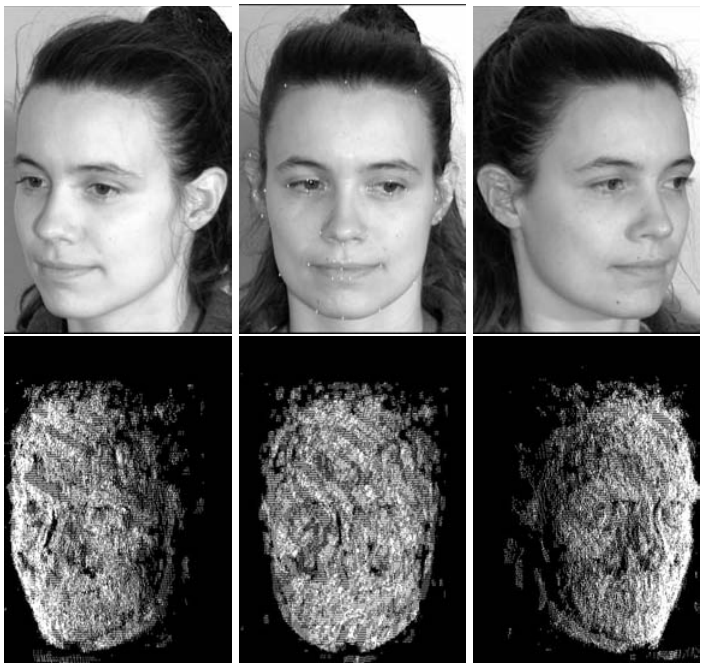


Figure 5.1: Uncalibrated video sequence: Top row: Three images from the uncalibrated video sequence out of eight images. In the middle image 2D manually provided feature points, where some of them are used for initialization, are depicted. Bottom Row: Data points obtained as centers of local surface patches fitted to the raw stereo data.

5.2 Silhouettes

When we talk about silhouettes we think about the contours in the image that represent projection of the object occluding contours. First we draw manually, an outline in whose proximity there is a silhouette. In the top row of Fig. 5.2 we depicted initially outlined contours. Such contours are used as an initial guess for already well known, active contour algorithm of [70]. After the convergence of the algorithm we obtained good outlines that represent the projection of the subject's occluding contour. Sometimes, as depicted in the right end image of Fig. 5.2, the snake algorithm diverges especially in the presence of the strong edges around the potential silhouettes. To remedy this problem, fine tuning of the snake algorithm parameters is required together with providing a new initial guess. This way of providing silhouettes might be justified in case of short video sequences. However, when we want to extensively use silhouettes for tracking of the nonrigid motion and for more precise and error prone silhouette detection, we need a more sophisticated mechanism.

Figure 5.2: Silhouette detection using active contours: Top row: Manually outlined initial guesses used as an input for snakes algorithm. Bottom row: Final silhouette contours after the snake algorithm converged. Note that in the last image snake was attracted by the strong edges of the background.

This can be provided, thanks to the implicit meshes, and will be addressed in the following chapter. In this chapter we will concentrate on fitting to the manually provided silhouette contours, that take advantage of differentiable properties of the implicit meshes.

Their usage of once detected 2–D silhouette contours, will heavily depend on the generic model and its properties. In case of the explicit meshes, one has to search for the silhouette facets on the model, and use them in the fitting framework. On the other hand, the implicit meshes provides more elegant way of handling silhouettes. We will discuss separately both contexts of using silhouettes, compare them and highlight their differences.

5.3 Objective Functions

The generic models, such as once of Fig. 3.2(a,c), can be used as explicit polygonal meshes or as implicit meshes. In order to demonstrate power of implicit mesh representations we compare fitting both representations to noisy data. Since we are using data sources that are different by its nature, such as stereo and silhouettes, we are going to associate separate objective functions corresponding to each type of the observation. However, since we want to put all of them together in a common fitting framework, we have to guarantee that their influences are commensurate. Generally, all our objective functions minimizes the squares of distances in a least-square sense. For that purpose we use Levenberg-Marquardt least square algorithm.

In this section we continue with objective functions for implicit and explicit meshes. We associate a particular objective function for each data type, including stereo and silhouette data.

5.3.1 Objective Function for Implicit Meshes

In the case of implicit meshes, we use the image data to write n_{obs} *observation equations* of the form

$$Obs(\mathbf{x}_j, \mathbf{S}) = T - F(\mathbf{x}_j, \mathbf{S}) = \epsilon_i \ , \ 1 \leq j \leq n_{obs} \ , \tag{5.1}$$

where F is the field function of either Eq. 4.14 or 4.15, $\mathbf{x}_j$ one of the data points, $\mathbf{S}$ the state vector of Eq. 4.12, T is a given isovalue, and ϵ_i the corresponding residual. ϵ_i is the algebraic distance to the implicit mesh and should be as small as possible. Fitting therefore implies minimizing

$$\chi^2 = \mathbf{Obs}^T(\mathbf{S}) W \mathbf{Obs}(\mathbf{S}) \tag{5.2}$$

where $\mathbf{Obs}(\mathbf{S}) = [\epsilon_1, \ldots, \epsilon_{n_{obs}}]$ is the vector of residuals and W a diagonal weight matrix associated to the observations. In practice, our system must be able to deal with observations coming from different sources. To guarantee that their influences are commensurate, we assign a weight w_{type_i} to each kind of observation, where $type_i$ is the nature of the observation, and minimize

$$\chi^2 = 1/2 \sum w_{type_i} \epsilon_i^2 \ , \tag{5.3}$$

where the w_{type_i} are chosen so that the contribution to the objective function gradients of all the observations of a particular kind are of similar magnitudes [47] and it is explained bellow.

5.3.1.1 Stereo Data

As already said, we use either a simple correlation-based technique [46] or a more sophisticated max-flow stereo algorithms [96] to compute potentially noisy clouds of 3–D points. Each one is used to produce one observation of the kind described by Eq. 5.1 and can be written as follows for spherical implicit meshes:

$$Obs^{stereo}(\mathbf{x}_j, \mathbf{S}) = T - F(\mathbf{x}_j, \mathbf{S}) = T - \sum_{i=1}^{N} f(d_i(\mathbf{x}_j, \mathbf{S}), r_i(\mathbf{S})) = \epsilon_j, 1 \leq j \leq nobs \tag{5.4}$$

where f is exponential field function of each primitive, $d_i(\mathbf{x}_j, \mathbf{S})$ is a Euclidean distance of the observation $\mathbf{x}_j$ to the implicit surface depending on the state vector $\mathbf{S}$, and $r_i(\mathbf{S})$ is a radius of the spherical metaball that also depend on the state vector $\mathbf{S}$ that is directly responsible for the surface shape. In case of the triangular implicit mesh the objective function can be written like:

$$Obs^{stereo}(\mathbf{x}_j, \mathbf{S}) = T - F(\mathbf{x}_j, \mathbf{S}) = T - \sum_{i=1}^{N} f(d_i(\mathbf{x}_j, \mathbf{S})) = \epsilon_j, 1 \leq j \leq nobs \tag{5.5}$$

Note that in this case the shape of the implicit surface does not depend on the size of the facets. Minimizing the corresponding residuals tends to force the fitted surface to be

as close as possible to these points. The distance function is an algebraic distance between the 3–D data points and the implicit part of the implicit mesh. Finally, the explicit part of the implicit mesh will be accepted as a final result, even thought it is placed on the distance equal to the average implicit mesh thickness. This is because, that thickness is chosen to be very small, thanks to the good approximation by the implicit part. In this way it is not necessary to additionally convert implicit surface for the rendering purposes, which significantly saves computation time necessary for visualization of the results. This is true for triangular implicit meshes, while for spherical implicit meshes it is different. To obtain the final result in this case, the explicit mesh inside has to be additionally extruded, so to align to the parallelly placed implicit envelope.

The properties of the chosen distance function allow the system to naturally deal with outliers and to converge even from rough initializations or estimates. The smooth shape of the inverted exponential that is used in our field function is responsible for both effects. Because it approaches zero asymptotically, distant data points have an influence, which helps if the initial position is inaccurate, but it is limited. As a result, outliers are naturally ignored because their contribution is dwarfed by that of inliers.

5.3.1.2 Silhouette Data

For each instance of the state vector $\mathbf{S} \in R$, we define the implicit surface as a part of implicit mesh like:

$$L(\mathbf{S}) = \left\{ \mathbf{x} \in R^3, F(\mathbf{x}, \mathbf{S}) = T \right\} \tag{5.6}$$

Given the line of sight $\mathbf{Sl}_j$ defined by a silhouette point $\mathbf{e}_j$ in the image and the camera center $\mathbf{c}$, let $\mathbf{x}_j$ be the point along this line of sight where it is tangential to $L(\mathbf{S})$. By definition, $\mathbf{x}_j$ satisfies two constraints

1. The point is on the surface, therefore $F(\mathbf{x}_j, \mathbf{S}) = T$;
2. The normal $\mathbf{n}_j(\mathbf{S}) = [\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}]$ to $L(\mathbf{S})$ is perpendicular to the line of sight $\mathbf{Sl}_j$ at $\mathbf{x}_j$, therefore $\mathbf{n}_j(\mathbf{S}) \bullet \mathbf{Sl}_j = 0$.

We integrate silhouette observations into our framework by performing an initial search along the line of sight $\mathbf{Sl}_j$ to find the point $\mathbf{x}_j$ that is closest to the model in its current configuration, which by construction satisfies the second constraint. We can distinguish three cases for the search along the line of sight depicted in the Fig. 5.3(a, b, c). In Fig. 5.3(a) line of sight does not intersect implicit surface at all, while in Fig. 5.3(b,c) it intersects implicit surface. Since the implicit surface is created as an envelope around the explicit mesh, it appears to be double-sided. In Fig. 5.3(b) line of sight completely passes two times thru both sides of the implicit surface. On the other hand in Fig. 5.3(c) goes only thru the outer side of the implicit surface. In order to speed up our search, we create the bounding box around the object, and find intersection of the lines of sights with the front and back planes of the bounding box. Then the search is performed between those intersection points. The search is simple evaluation of the values of the potential field function of sampled points

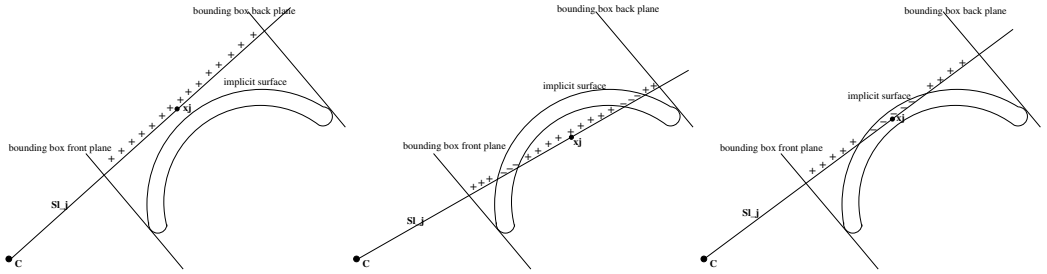


Figure 5.3: Silhouette point search for the implicit mesh: (a) Line of sight does not intersect implicit surface. (b) Line of sight completely passes two times thru both sides of the implicit surface. (c) Line of sight goes only thru the outer side of the implicit surface.

that lay on the line of sight and are distributed at the certain step distance one from each other. Potential field function will return positive values for the points outside the implicit mesh, whose values will approach zero when they approach the iso-surface of the implicit surface, and then become negative when they are inside of the implicit surface. There are two ways of selecting the closest point to the model: one applies to the cases of Fig. 5.3(a,b) and the point $\mathbf{x}_j$ with the smallest positive potential filed value is selected, and the other applies to the case of Fig. 5.3(c) where the point with the biggest negative potential field value will be selected.

This selected point is used to add one of the observations described by Eq. 5.1 and minimizing the corresponding residual tends to enforce the first constraint, in the same manner as it is done for the stereo observations and is represented like:

$$Obs^{silh}(\mathbf{x}_j, \mathbf{S}) = T - F(\mathbf{x}_j, \mathbf{S}) = T - \sum_{i=1}^{N} f(d_i(\mathbf{x}_j, \mathbf{S})) = \epsilon_j, 1 \leq j \leq nobs \qquad (5.7)$$

Note that, as the model changes shape, $\mathbf{x}_j$ must move along the ray to remain the point that is closest to the model. During optimization, this must be taken into account when evaluating the derivatives of the residual because the $\mathbf{x}_j$ term has now become a function of $\mathbf{S}$, and has to be expressed as $\mathbf{x}_j = \mathbf{x}_j(\mathbf{S})$. As discussed bellow, this involves evaluating the first and second order derivatives of F.

To exploit this constraints in a least-square context one must compute:

$$\frac{\partial F(x(\mathbf{S}), y(\mathbf{S}), z(\mathbf{S}), \mathbf{S})}{\partial s} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial s} + \frac{\partial F}{\partial s}, s \in \mathbf{S} \qquad (5.8)$$

which requires the computation of $\frac{\partial x}{\partial s}$, $\frac{\partial y}{\partial s}$ and $\frac{\partial z}{\partial s}$. These can be derived as follows. The line of sight is tangential to the surface at , therefore

$$\left(\frac{\partial F(\mathbf{x}(\mathbf{S}),\mathbf{S})}{\partial x},\frac{\partial F(\mathbf{x}(\mathbf{S}),\mathbf{S})}{\partial y},\frac{\partial F(\mathbf{x}(\mathbf{S}),\mathbf{S})}{\partial z}\right)\bullet\mathbf{Sl}_j=0$$

Differentiating that with respect to $\mathbf{S}$ yields:

$$\begin{aligned}0=&(\tfrac{\partial^2 F}{\partial x\partial x},\tfrac{\partial^2 F}{\partial x\partial y},\tfrac{\partial^2 F}{\partial x\partial z})\bullet\mathbf{Sl}(\mathbf{S})\tfrac{\partial x}{\partial s}\\&+(\tfrac{\partial^2 F}{\partial x\partial y},\tfrac{\partial^2 F}{\partial y\partial y},\tfrac{\partial^2 F}{\partial y\partial z})\bullet\mathbf{Sl}(\mathbf{S})\tfrac{\partial y}{\partial s}\\&+(\tfrac{\partial^2 F}{\partial x\partial z},\tfrac{\partial^2 F}{\partial y\partial z},\tfrac{\partial^2 F}{\partial z\partial z})\bullet\mathbf{Sl}(\mathbf{S})\tfrac{\partial z}{\partial s}\\&+(\tfrac{\partial^2 F}{\partial x\partial s},\tfrac{\partial^2 F}{\partial y\partial s},\tfrac{\partial^2 F}{\partial y\partial s})\bullet\mathbf{Sl}(\mathbf{S})\end{aligned}\tag{5.9}$$

Furthermore, $\mathbf{x}(\mathbf{S})=[x(\mathbf{S}),y(\mathbf{S}),z(\mathbf{S})]^T$is constrained to move along the line of sight, therefore:

$$N1_x\frac{\partial x}{\partial s}+N1_y\frac{\partial y}{\partial s}+N1_z\frac{\partial y}{\partial s}=0\tag{5.10}$$

$$N2_x\frac{\partial x}{\partial s}+N2_y\frac{\partial y}{\partial s}+N2_z\frac{\partial y}{\partial s}=0\tag{5.11}$$

where $\mathbf{N1}=[N1_x,N1_y,N1_z]^T$ and $\mathbf{N2}=[N2_x,N2_y,N2_z]^T$ are two additional vectors such that $\mathbf{N1}$, $\mathbf{N2}$, $\mathbf{Sl}_j$ form an orthogonal referential. Equations 5.9, 5.10, 5.11 are three linear equations in three unknowns $\frac{\partial x}{\partial s}$, $\frac{\partial y}{\partial s}$ and $\frac{\partial z}{\partial s}$ that can thus be computed. $\frac{\partial F(x(\mathbf{S}),y(\mathbf{S}),z(\mathbf{S}),\mathbf{S})}{\partial s}$ can then be derived using the chain rule of Eq. 5.8.

This approach to taking silhouette information into account does not require us to search for specific facets and imposes no restriction on the topology and complexity of the triangulated model we use. We will see below that such is not the case when using explicit meshes as opposed to implicit ones. However, such search is computationally expensive since it has to be performed after each iteration step of one optimization. In the next chapter we will discuss how implicit meshes can be used for efficient silhouette detection, and facilitate their use for monocular tracking and 3–D reconstruction.

5.3.2 Objective Functions for Explicit Meshes

For comparison's sake, we have also implemented an objective function that lets us fit an explicit mesh to the same stereo and silhouette data, but without using the implicit surface formalism we advocate. This objective function is similar to the one of Eq. 5.1, except for the fact that we must compute differently the ϵ_j residuals that appear in the χ^2 term of Eq. 5.3. As in Section 5.3.1, we distinguish between stereo and silhouette data, so here, we will go down the same line.

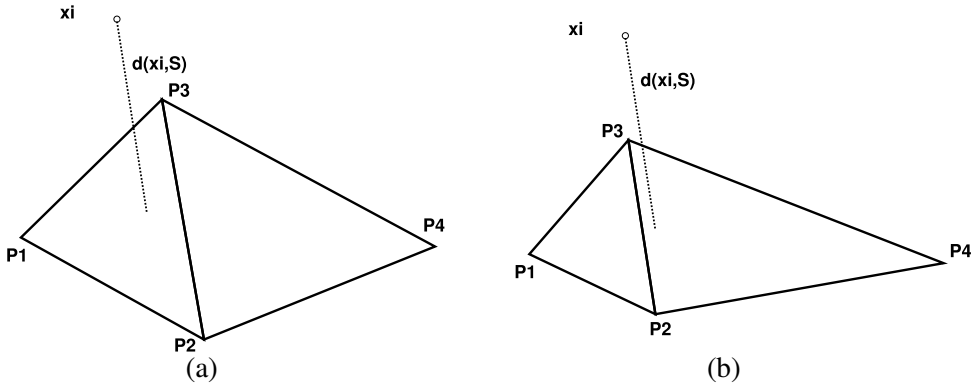


Figure 5.4: Non-differentiability of the distance function used to fit an explicit mesh to a cloud of 3–D points. (a) A data point $\mathbf{x}_j$ is initially closest to the P_1, P_2, P_3 facet. (b) After a certain number of iterations, the mesh has deformed and $\mathbf{x}_j$ is now closer to the P_2, P_3, P_4 facet. Accounting for this change in the objective function results in non-differentiability.

5.3.2.1 Stereo Data

For a 3–D data point $\mathbf{x}_j$, we replace the algebraic distance of Eq. 5.4 or Eq. 5.5 by the orthogonal distance to the "closest" facet

$$Obs^{ster}(\mathbf{x}_j, \mathbf{S}) = \frac{\mathbf{n}(\mathbf{S}) \bullet (\mathbf{x}_j - \mathbf{p}(\mathbf{S}))}{\|\mathbf{n}(\mathbf{S})\|} = \epsilon_j^{stereo} \quad , \tag{5.12}$$

where $\mathbf{n}(\mathbf{S})$ is a vector normal to the facet and $\mathbf{p}(\mathbf{S})$ one of its vertices.

Here we take the closest facet to be one of those that defines a triangular cylinder that contains $\mathbf{x}_j$ and, if no such facet exists, we simply ignore the data point. As shown in Fig. 5.4, as the optimization progresses and the facets move, the closest facet may change, which introduces non-differentiabilities if taken into account.

5.3.2.2 Silhouette Data

Given a silhouette point, we look for a triangulation facet that is almost parallel to the line of sight it defines and such that there is a 3–D point along this line for which the distance of Eq. 5.12 is small. If such a facet exists, it projects almost edge-on and is therefore likely to produce an occluding contour that goes through the silhouette point.

Let $\mathbf{Sl}$ be the unit vector that represents the direction of the line of sight. To enforce the silhouette constraint, we search for the facet whose normal is almost perpendicular to $\mathbf{Sl}$, that is such that $|\mathbf{n} \bullet \mathbf{Sl}| \leq \zeta$ where ζ is a small constant, and along which there is a point $\mathbf{x}$

such that the distance of Eq. 5.12 is smallest. We use this facet and this point to define two residuals

$$\begin{aligned} Obs^{silh}(\mathbf{x},\mathbf{S}) &= \frac{\mathbf{n}(\mathbf{S}) \bullet (\mathbf{x} - \mathbf{p}(\mathbf{S}))}{\|\mathbf{n}(\mathbf{S})\|} = \epsilon^{silh} \\ Obs^{norm}(\mathbf{x},\mathbf{S}) &= \mathbf{Sl} \bullet \mathbf{n}(\mathbf{S}) = \epsilon^{norm} \end{aligned}$$

whose weighted sum of squares replace the corresponding silhouette terms in Eq. 5.3.

In addition to the non-differentiabilities that looking for the "closest" facet introduces and that we discussed above, the main problem with this formulation is that these residuals are critically dependent on mesh resolution and regularity. If the facets are irregular or large where the surface slants away from the camera, it will be difficult to find appropriate "silhouette" facets. This results in one of the poor results shown in Chapter 7, that are corrected by replacing the explicit meshes by implicit ones.

5.4 Weighting Observations

As already mentioned, our system must be able to deal with many observations coming from different sources that may not be commensurate with each other. Formally we minimize the the weighted chi square error of Eq. 5.3. To ensure that these matrices are well conditioned and that the minimization proceeds smoothly we multiply the weight w_{type_i} of the n_{type} individual observations of a given type by a global coefficient c_{type}computed as follows:

$$\begin{aligned} G_{type} &= \frac{\sqrt{\sum_{1\leq j\leq nobs, i=type} w_{type_j} \|\nabla Obs^i(\mathbf{x}_j,\mathbf{S})\|^2}}{n_{type}} \\ c_{type} &= \frac{\lambda_{type}}{G_{type}} \end{aligned} \tag{5.13}$$

where λ_{type} is a user supplied coefficient between 0 and 1 that indicates the relative importance of the various kinds of observations. This guarantees that, initially at least, the magnitudes of the gradient terms for the various types have the appropriate relative values.

5.5 Attaching Observations

Before fitting is started, data points coming either from stereo or silhouettes are associated to the generic surface model. The way of attaching them to the model depends on the choice of the model and will be separately explained for the implicit meshes and for the explicit meshes.

5.5.1 Explicit Mesh Attachment

In case of explicit meshes it is important to associate each facet to its closest observation point. The brute force approach would be to check each data point against all the facets, that

is very slow. Instead we create a bounding box around the data and the surface model, and voxelize it by subdivision in all three directions. Then to each voxel we associate the list of facets, whose centers of the gravity are contained inside. After we take every data point, find the voxel to which it belongs, and traverse the list of associated facets of that voxel to verify to which of those facets, the data point normally projects and is on the minimal distance from it. In this way, every data point will end up to have a closes facet, to which it projects, attached to it. This approach dramatically shortens the search for the closest facets.

5.5.2 Implicit Mesh Attachment

Remember, that the implicit mesh is created by associating volumetric primitives, such as spherical or triangular metaballs to each triangular facet of the mesh. Also, each metaball creates scalar potential field around it that has some influence. The iso-surface taken around one metaball collects all the points in space where their potential is zero, but going further from that iso-surface the potential field exponentially decreases until it reaches the given threshold t of Eqs. 4.7, 4.1. In order to speed up the computation, we then associate to each data point, those metaballs, i.e. facets, which influence that data point. It happens that most of the metaballs in the proximity are attached, while the great majority that is far away is not considered. The re-attaching is performed after each minimization. This is valid for both triangular and spherical metaballs.

5.6 Regularization

Because there are both noise and gaps in the image data, we still found it necessary to introduce a regularization term. This term depends on the underlying model that is used to parameterize surface model representation that is used to represent the model of the object we want to track or reconstruct. We have been using three different model parameterizations including: DFFD transformation for upper body modeling and tracking, PCA based statistical model for face reconstruction and directly surface model vertices for tracking paper deformation. Each of them involves different regularization terms, which smooth overall deformation. Bellow, we discuss those regularization and deformation energies they add to the main energy function which we minimize.

5.6.1 DFFD Regularization

Since, we expect the deformation between the initial shape and the original one to be smooth, this can be done by preventing deformations at neighboring vertices of the control mesh to be too different. This is enforced by introducing a deformation energy E_D that approximates the sum of the square of the derivatives of displacements across the control surface. By treating the control triangulation facets as C^0 finite elements, we write

$$E_D = \Delta_x^t K \Delta_x + \Delta_y^t K \Delta_y + \Delta_z^t K \Delta_z \tag{5.14}$$

where K is a stiffness matrix and Δ_x, Δ_y and Δ_z are the vectors of the *x, y* and *z* coordinates of the control vertices' displacements. The term we actually optimize becomes:

$$E = \sum_{i=nobs}^{nobs} w_{type_i} \epsilon_i^2 + \lambda_D E_D$$

where λ_D is a small positive constant. In the Section 3.2.4 this is explained in details.

5.6.2 PCA Regularization

In case of face modeling where PCA face model [7] is used a small regularization term is added to the total energy function. The function we actually minimize is therefore:

$$E = \frac{\sum_{i=nobs}^{nobs} w_{type_i} \epsilon_i^2}{\sigma_N^2} + \sum_{i=1}^{99} \frac{\alpha_i^2}{\sigma_{S_i}^2} \tag{5.15}$$

where the σ_{S_i} are the eigen values of the shape covariance matrix provided with the model [7] and σ_N an initially large constant that progressively decreases. α_i are PCA coefficients defining the state vector $\mathbf{S}$.

5.6.3 Triangular Mesh Regularization

Tracking deformable piece of paper involves a simple planar triangulation used as a model. In this case the state vector $\mathbf{S}$ is composed of mesh vertices. To keep the deformations physically plausible, we define a deformation energy that is the sum of two terms. The first one represents the inextensibility of the paper by penalizing the variations of the distance between a vertex and its neighboring vertices, and the second one models the bending stiffness of paper by constraining the curvature of the mesh that can be written as follows:

$$E_D = \sum_{i=1}^{Nvert} \Big(\sum_{v_j \in Neighbors(v_i)} w_{ext} (\|v_i - v_j\| - L)^2 + \tag{5.16}$$

$$+ \sum_{(v_j, v_k) \in Neighbors(v_i)} w_{bend} \|2v_i - v_j - v_k\|^2 \Big) \tag{5.17}$$

where v_j is one vertex of the mesh, $Neighbors(v_i)$ represents the set of all its neighbors, L is a given initial edge length equal for the whole mesh and w_{ext} and w_{bend} are two user defined weights.

5.7 Objective Function Derivatives

Since we concentrate on the implicit meshes, especially on the triangular implicit meshes, we are going to present the derivatives computation of the objective functions concerning matrix representation of the triangular metaballs. This representation appears to be compact and elegant way of expressing function derivatives. For the sake of comparison, the derivative computation when the matrix notation is not used, will be also outlined.

5.7.1 Derivatives Without Matrix Notation

We will consider just derivatives of potential filed function of one triangular metaball expressed by the Eq. 4.7. The derivatives of overall potential filed function is obtained by summation of all individual potential field derivatives. First we compute the derivatives of the potential filed function, which is complex function depending on the distance function $d(\mathbf{x}, \mathbf{S})$ and state vector parameters $\mathbf{S}$:

$$\frac{\partial f}{\partial s} = -k\frac{\partial d(\mathbf{x}, \mathbf{S})}{\partial s}\exp^{-k(d(\mathbf{x},\mathbf{S})-d_0^2)}, s \in \mathbf{S}$$

Now, we have to compute derivatives of the distance function of Eq. 4.6, which is composed of seven distances. We will only consider the derivatives of the distance from the points to plane, while the others, such as distance point to line and point to point are similarly computed. The facet normal is common for all distance functions, which depends on the facet vertices, and consequently on the model parameters:

$$\mathbf{n}(\mathbf{S}) = [n_x(\mathbf{S}),\, n_y(\mathbf{S}),\, n_z(\mathbf{S})]^T = (P_2(\mathbf{S}) - P_1(\mathbf{S})) \times (P_3(\mathbf{S}) - P_1(\mathbf{S}))$$

In order to compute other derivatives, one have to compute derivatives of the normal in respect to the model parameters:

$$\frac{\partial \mathbf{n}(\mathbf{S})}{\partial s} = (P_2(\mathbf{S}) - P_1(\mathbf{S})) \times \frac{\partial (P_3(\mathbf{S}) - P_1(\mathbf{S}))}{\partial s} + \frac{\partial (P_2(\mathbf{S}) - P_1(\mathbf{S}))}{\partial s} \times (P_3(\mathbf{S}) - P_1(\mathbf{S}))$$

Further, derivative of the distance point to plane are computed as follows:

$$\frac{\partial d(\mathbf{x}, \mathbf{S})}{\partial s} = 2\frac{\mathbf{n}(\mathbf{S}) \bullet (x - P_1(\mathbf{S}))}{\|\mathbf{n}(\mathbf{S})\|}\left(\frac{\frac{\partial (\mathbf{n}(\mathbf{S}) \bullet (x - P_1(\mathbf{S}))}{\partial s}\|\mathbf{n}(\mathbf{S})\| + (\mathbf{n}(\mathbf{S}) \bullet (x - P_1(\mathbf{S})))\frac{\partial \|\mathbf{n}(\mathbf{S})\|}{\partial s}}{\|\mathbf{n}(\mathbf{S})\|^2}\right)$$

where the derivatives of the normal vector's norm is:

$$\frac{\partial \|\mathbf{n}(\mathbf{S})\|}{\partial s} = \frac{n_x(\mathbf{S})\frac{\partial n_x(\mathbf{S})}{\partial s} + n_y(\mathbf{S})\frac{\partial n_y(\mathbf{S})}{\partial s} + n_y(\mathbf{S})\frac{\partial n_y(\mathbf{S})}{\partial s}}{\sqrt{n_x(\mathbf{S})^2 + n_y(\mathbf{S})^2 + n_z(\mathbf{S})^2}}$$

Fitting to silhouettes requires also computation of the first and second order derivatives in respect to the point coordinates and parameters. So, the derivatives of the potential field in respect to the *x* coordinate are:

$$\begin{aligned}\frac{\partial f}{\partial x} &= \frac{\partial f}{\partial d}\frac{\partial d}{\partial x} = -k\exp^{-k(d(\mathbf{x},\mathbf{S})-d_0^2)}\frac{\partial d}{\partial x}\\ \frac{\partial^2 f}{\partial x \partial x} &= -k\exp^{-k(d(\mathbf{x},\mathbf{S})-d_0^2)}\left(\frac{\partial^2 d}{\partial x \partial x} - k\left(\frac{\partial d}{\partial x}\right)^2\right)\end{aligned}$$

The other derivatives that has to be computed are:

$$\frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial x \partial z}, \frac{\partial^2 f}{\partial y \partial x}, \frac{\partial^2 f}{\partial y \partial y}, \frac{\partial^2 f}{\partial y \partial z}, \frac{\partial^2 f}{\partial z \partial x}, \frac{\partial^2 f}{\partial z \partial y}, \frac{\partial^2 f}{\partial z \partial z}$$

$$\frac{\partial^2 d}{\partial x \partial x}, \frac{\partial^2 d}{\partial x \partial y}, \frac{\partial^2 d}{\partial x \partial z}, \frac{\partial^2 d}{\partial y \partial x}, \frac{\partial^2 d}{\partial y \partial y}, \frac{\partial^2 d}{\partial y \partial z}, \frac{\partial^2 d}{\partial z \partial x}, \frac{\partial^2 d}{\partial z \partial y}, \frac{\partial^2 d}{\partial z \partial z}$$

Note that the overall complexity of the derivatives computation is increasing. If we use matrix notation, as it will be presented bellow, this is much easier to handle.

5.7.2 Derivatives Under Matrix Notation

In this section we will relay on the Eqs. 4.11, representing distance function in a matrix form of the triangular metaball. Derivatives of the potential field function are the same as explained above, while the distance function derivatives computation becomes far more easier. First we will show you the first order derivatives of the distance function in respect to the model parameters:

$$\begin{aligned}\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s} &= \mathbf{x}^T\frac{\partial L_1(\mathbf{S})}{\partial s}L_2\mathbf{x} + \mathbf{x}^T L_1\frac{\partial L_2(\mathbf{S})}{\partial s}\mathbf{x}\\ \frac{\partial d_c(\mathbf{x},\mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S})L_c\frac{\partial T_r(\mathbf{S})}{\partial s}\mathbf{x}\\ \frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S})L_s\frac{\partial T_r(\mathbf{S})}{\partial s}\mathbf{x}\end{aligned}$$

while first order derivatives over independent variables x, y and z are:

$$\begin{aligned}\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial x_i} &= \frac{\partial \mathbf{x}^T}{\partial x_i}L_1L_2\mathbf{x} + \mathbf{x}^T L_1L_2\frac{\partial \mathbf{x}}{\partial x_i} = 2L_1L_2\frac{\partial \mathbf{x}}{\partial x_i}, \quad L_1L_2 = (L_1L_2)^T\\ \frac{\partial d_c(\mathbf{x},\mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S})L_cT_r\frac{\partial \mathbf{x}}{\partial x_i}\\ \frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S})L_sT_r\frac{\partial \mathbf{x}}{\partial x_i}\end{aligned}$$

Consequently, second order derivatives over coordinates are:

$$\frac{\partial^2 d_c}{\partial x_i \partial x_j} = 2\frac{\partial \mathbf{x}^T}{\partial x_j} T_r^T L_c T_r \frac{\partial \mathbf{x}}{\partial x_i}$$

$$\frac{\partial^2 d_s}{\partial x_i \partial x_j} = 2\frac{\partial \mathbf{x}^T}{\partial x_j} T_r^T L_s T_r \frac{\partial \mathbf{x}}{\partial x_i}$$

$$\frac{\partial^2 d_p}{\partial x_i \partial x_j} = 2\frac{\partial \mathbf{x}^T}{\partial x_j} L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i}$$

and over the parameters are:

$$\frac{\partial d_c}{\partial x_i \partial s} = 4\mathbf{x}^T T r^T L_c \frac{\partial T_r}{\partial s} \frac{\partial x}{\partial x_i}$$

$$\frac{\partial d_s}{\partial x_i \partial s} = 4\mathbf{x}^T T r^T L_s \frac{\partial T_r}{\partial s} \frac{\partial x}{\partial x_i}$$

$$\frac{\partial d_p}{\partial x_i \partial s} = 2\mathbf{x}^T \left(\frac{\partial L_1}{\partial s} L_2 + L_1 \frac{\partial L_2}{\partial s} \right) \frac{\partial x}{\partial x_i}$$

where everywhere $\frac{\partial \mathbf{x}}{\partial x_i} = [x_1, x_2, x_3, 0]^T, x_i = 1, x_{k \neq i} = 0$. Note the compactness of the expression obtained using this matrix representation. Finally, in order to speed up overall computation, it is possible to compute distance using non-matrix notation, since matrix multiplication seems to be expensive for distance computation, and use matrix notation for computing derivatives that appears to be faster then direct derivatives computation as shown in the previous section.

5.8 Summary

In this chapter we have given our optimization framework that takes advantage of stereo and silhouette data, and compares implicit meshes against explicit surface representations. We would like to stress efficacy of silhouette fitting in case of implicit meshes, where differentiability of our surface representation delivered mathematically elegant framework of handling silhouettes. We also showed the explicit surface weakness in terms of distance function non-differentiability and silhouette handling. In the end we compared, brute force distance function derivatives computation in case of using directly Euclidean distances, and the elegant derivatives computation in case of using matrix formulation of the distance function.

6 Implicit Meshes Make for Better Silhouettes

Occluding contours are a key clue to recovering the shape of smooth and potentially deformable surfaces in monocular sequences and they have been used extensively for this purpose. However, because extracting them reliably against potentially cluttered or changing backgrounds such as those of Fig. 6.1, is difficult, most of the published work involves engineering the environment to make this task easier. In a previous chapter we demonstrated how implicit meshes take advantage of occluding contours in sense of least-square optimization. However, silhouettes were detected using manually initialized active contour models, and finding corresponding object's occluding contours required expensive search along the lines of sights.

In this chapter, we show that representing generic 3-D surfaces as implicit meshes allows us to automatically detect silhouettes in the images and to take advantage of occluding contour constraints in more robust way. Furthermore, it also lets us effectively combine silhouette information with that provided by interest points that can be tracked from image to image. This is important because this may mean the difference between the ability or the inability to exploit silhouettes in uncontrolled real-world situations where occlusions and difficult backgrounds often degrade the output of even the best edge detection algorithms.

More specifically, the *implicit meshes* [61], allows us to robustly detect the occluding contours on the 3-D surface as the solution of an ordinary differential equation [95]. Their projections can then be used to search for the true image boundaries and deform the 3–D model so that it projects correctly. This well-formalized approach yields a robust implementation that we demonstrate for monocular tracking of deformable 3–D objects in a completely automated fashion: We start with a generic 3-D model of the target object, find its occluding contours, and use them to search for the corresponding contours in the images. We then use the detected 2-D contours and the constraints they impose, along with some feature information when available, to deform the model. On this procedure we based our tracking algorithm that successively fits the deformable objects following nonrigid forms of the objects.

This approach is effective independently of the specific way the deformations are parametrized. As shown in Fig.6.1, we validated the tracker in several very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations [84]; reconstructing the shape of a human face parametrized in terms of a Principal Component Analysis model [7, 34].

In the remainder of the chapter it is shown how we use implicit meshes first to guide the

search for silhouettes in the images, and second to enforce the corresponding differential constraints on the surface. We also address the use of feature points in our fitting framework, that can be combined with new silhouette fitting approach explained here.

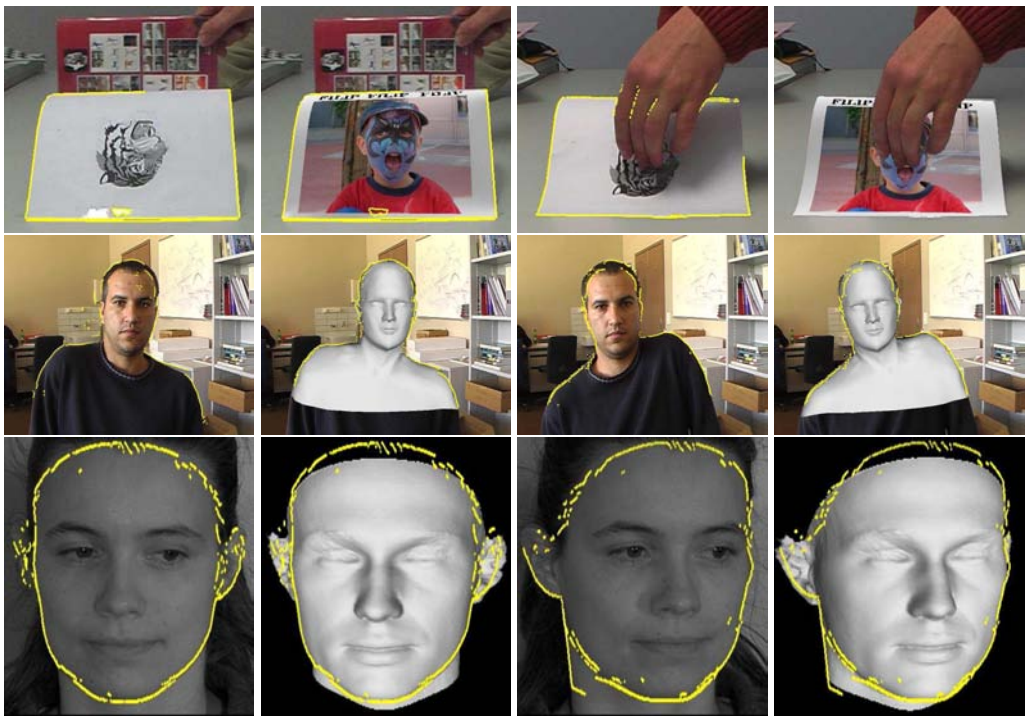
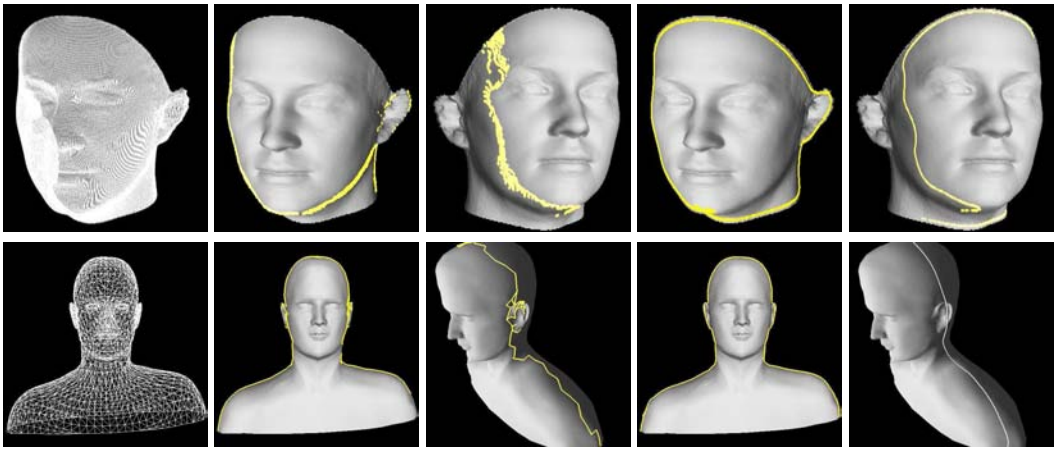


Figure 6.1: Detecting and using silhouettes for tracking and reconstruction from monocular sequences. The detected silhouette points are shown in yellow, or white if printed in black and white. First row: Tracking a deforming piece of paper with a tiger on it and replacing the tiger by a picture, which involves accurate 3–D shape estimation. This is done in spite of the moving book and the occluding hand. Middle row: Tracking the head and shoulders of a moving person. The reprojected 3–D model is shown as a shaded surface. Note that, even though the background is cluttered, we did not need to perform any kind of background subtraction. Bottom row: Precise reconstruction of a face from a short sequence in which the subject faces the camera.

6.1 Silhouette Detection

Remember, that previously, the silhouettes were detected directly in the image using snakes algorithm and the manual outlines. However, we cannot relay on such way of silhouette detection for tracking of deformable objects in monocular sequences, since, on one hand, it involves manual interventions and, on the other hand, very slow search along the line



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

Figure 6.2: Occluding contours on explicit versus implicit meshes. (a) High resolution mesh of the face and low resolution mesh of the upper body. (b) Shaded model with edges at the boundary between visible and hidden facets overlaid in yellow. (c) The same edges seen from a different viewpoint (d,e) Shaded models with the occluding contour computed using implicit mesh, corresponding to views (b) and (c) respectively. Note the much greater smoothness and improved precision.

of sights for the real occluding contours. To automatize this process one has to use the presence of the 3–D model to compute its 3–D occluding contours, project them into the image and use that projection as a starting guess to find the corresponding image boundaries, which should be the real silhouettes. This approach is essentially different then the previous one, since we go from the model's occluding contour and use that as a clue for silhouette detection in the image.

In this section, we first show some of the problems involved in performing this task using traditional techniques. We then show that our implicit mesh formalism solves them and gives us cleaner and more consistent results, which can then be exploited to detect the right image boundaries.

6.1.1 Occluding Contours from Explicit Meshes

In the absence of the implicit surface formalism we propose, one of the most popular ways of finding occluding contours is to perform a visibility computation: For example, we can use OpenGL to project the model into the images and flag the hidden facets. The edges at the border between visible and invisible facets whose normals satisfy the appropriate constraints can then be treated as candidate occluding contours.

As shown in Fig. 6.2(b,c), the results of this procedure are heavily dependent on mesh resolution and the resulting contours are rarely as smooth as they should. Of course, more

sophisticated heuristics would certainly yield improved results but we are not aware of any existing technique whose results are as clean and mesh-resolution independent as those of Fig. 6.2(d,e), which were obtained using our implicit surface formalism.

6.1.2 Occluding Contours and Ordinary Differential Equations

As shown in [95], occluding contours of implicit surfaces can be found by solving an ordinary differential equation (ODE) as follows: Let $\mathbf{x}(t)$, $t \in [0,1]$ be a 3–D occluding contour on the implicit surface $L(\mathbf{S})$ of Eq. 4.14, such as the one depicted by Fig. 6.3. For all values of t,

1. $\mathbf{x}(t)$ is on the surface and therefore $F(\mathbf{x}(t), \mathbf{S}) = T$,
2. the line of sight is tangential to the surface at $\mathbf{x}(t)$, and can be written as $(\mathbf{x}(t) - \mathbf{COpt}) \bigtriangledown F(\mathbf{x}(t)) = 0$.

Differentiating upper equations in respect to the parameter t we obtain:

$$\frac{\partial F(\mathbf{x}(t))}{\partial \mathbf{x}} \frac{\partial \mathbf{x}(t)}{\partial t} \quad = \bigtriangledown F(\mathbf{x}(t)) \frac{\partial \mathbf{x}(t)}{\partial t} = \quad 0 \tag{6.1}$$

$$\frac{\partial \mathbf{x}(t)}{\partial t} \bigtriangledown F(\mathbf{x}(t)) + (\mathbf{x} - \mathbf{COpt}) H(\mathbf{x}(t)) \frac{\partial \mathbf{x}(t)}{\partial t} = 0 \tag{6.2}$$

Replacing Eq. 6.1 into Eq. 6.2, and having in mind orthogonality of the vectors $\frac{\partial \mathbf{x}(t)}{\partial t}$, $\bigtriangledown F(\mathbf{x}(t))$, $(\mathbf{x} - \mathbf{Copt}) H(\mathbf{x}(t))$, this implies, similar to [95] that $\mathbf{x}(t), t \in [0,1]$ is a solution of the ODE:

$$\frac{\partial \mathbf{x}(t)}{\partial t} = \frac{(H(\mathbf{x}(t))(\mathbf{x}(t) - \mathbf{COpt})) \times \bigtriangledown F(\mathbf{x}(t), \mathbf{S})}{\|(H(\mathbf{x}(t))(\mathbf{x}(t) - \mathbf{COpt})) \times \bigtriangledown F(\mathbf{x}(t), \mathbf{S})\|} \tag{6.3}$$

where $H(\mathbf{x}(t))$ is the Hessian matrix of F, $\bigtriangledown F(\mathbf{x}(t))$ its gradient vector and $\mathbf{COpt}$ the optical center of the camera, as shown in Fig. 6.3.

Solving this ODE requires an appropriate starting point $\mathbf{x}(0)$, that is one 3–D point on the occluding contour. To find one *single* vertex of the explicit mesh that is very likely to be an occluding vertex, we use a visibility algorithm similar to the one described in Section 6.1.1. We then project it onto the implicit mesh and search in the neighborhood of the projection for a point that satisfies the two above stated constraints. Note that this is very different from the approach of Section 6.1.1 because, since we only need one 3–D point, we can impose very tight constraints and thus ensure that it really is on the occluding contour. This results in the very clean contours of Fig. 6.2(d,e) that are quite insensitive to the resolution of the mesh used to compute them.

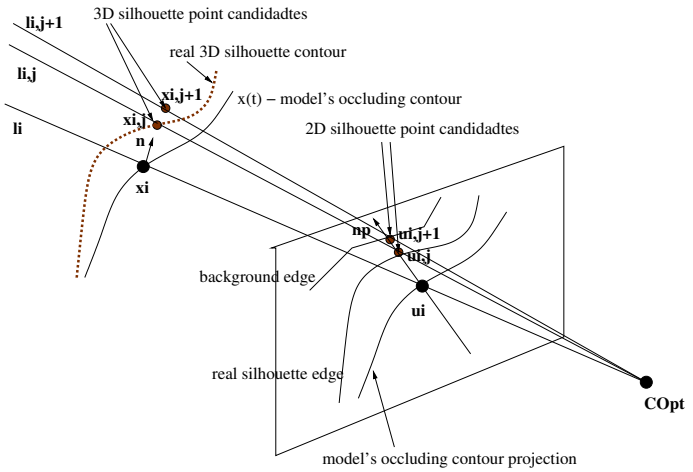


Figure 6.3: Finding multiple silhouette edge points in the image. Notations are defined in Section 6.1.3

6.1.3 Finding Silhouette Edges in the Image

Given a 3–D occluding contour $\mathbf{x}(t)$ computed as described above, we project it into the image and look for the true silhouette edge in a direction normal to its 2–D projection as depicted in Fig. 6.3. This is geometrically consistent because, at a silhouette point $\mathbf{x}_i \in \mathbf{x}(t)$, $t \in [0,1]$, the 3–D surface normal $\mathbf{n}$ is perpendicular to the line of sight $\mathbf{l}_i$ and, as a result, projects to the normal $\mathbf{n}_p$ of the 2–D contour.

In other words, at each point $\mathbf{u}_i$ of the 2–D projection, we simply have to perform a 1–D search along a scan-line for the true edge location and we are back to the old edge detection problem, but in a much simpler context than usual. We use a technique that has proved effective for edge-based tracking [121, 37]: Instead of selecting one arbitrary gradient maximum along the scan-line, we select multiple gradient maxima resulting in several potential silhouette edge points $\mathbf{u}_i^j$ and corresponding lines of sight $\mathbf{l}_i^j$ for each $\mathbf{x}_i$. Along these new lines of sight, we could choose the $\mathbf{x}_i^j$ where the line is closest to the surface as the most likely point where the surface should be tangent to the line of sight. However, this involves a computationally expensive search along the line of sight. In practice, as shown in Fig. 6.3, a simpler and equally effective approach is to take each $\mathbf{x}_i^j$ to be the point on $\mathbf{l}_i^j$ that is at the same distance from the optical center as the original $\mathbf{x}_i$. These $\mathbf{x}_i^j$ are then used as silhouette observations, as explained in Section 5.3.1.2.

Silhouettes are a key clue to surface shape and deformation in monocular sequences, but they are also a sparse one since they are only available at a few image locations. For objects that are somewhat textured, point correspondences between interest points in pairs of images complement them ideally. They can be established best where silhouettes are least useful, that is on the parts of the surfaces that are more or less parallel to the image plane.

In this section, we show that our formalism allows us to effectively combine these two information sources. Given a set of correspondences and silhouette points, we fit our model

to the data by minimizing a set of *observation equations* in the least-squares sense. To this end we use the Levenberg-Marquardt algorithm and, at each iteration, we recompute the occluding contours and corresponding silhouette points in the image using the technique of Section 6.1.

As we will see, the silhouette-based constraints are best expressed in terms of the implicit surface formalism while it is simpler to formulate the correspondence-based ones using traditional triangulated meshes. Recall from Section 4.1 that both the implicit mesh and the underlying explicit one deform in tandem when the state vector changes. As a result, we can simultaneously use the implicit formalism when dealing with silhouettes and the explicit one when dealing with correspondences as needed to simplify our implementation. We view this as one of the major strengths of our approach.

6.2 Least Squares Framework

We use the image data to write $n_{\rm obs}$ observation equations of the form given by Eq. 5.1. We now turn to the description of these functions for the two data types we use silhouettes and feature points.

6.2.1 Silhouettes

In Section 6.1, we showed how to use our formalism to associate 2–D image locations to 3–D surface points that lie on the occluding contours. If the shape and pose of the 3–D model were perfect, the 3–D points would project exactly at those locations. In other words, for each *i*, at least one of the candidate occluding points $\mathbf{x}_i^j$ introduced at the end of Section 6.1.3 should be on the surface, as shown in Fig. 6.3. During the optimization, this will in general not be true and we enforce this constraint by introducing a *silhouette function* of the form

$$Obs^{silh}(\mathbf{x}_i^j, \mathbf{S}) = w_i^j (F(\mathbf{x}_i^j, \mathbf{S}) - T), \tag{6.4}$$

for each $\mathbf{x}_i^j$, where w_i^j is the weight associated to the candidate, F the field function of Eq. 4.14, and T the isovalue defined in the same equation.

For each $\mathbf{x}_i^j$, w_i^j is taken to be inversely proportional to its distance to the line of sight $\mathbf{l}_i$. As a result, for each i only one of these candidates, $\mathbf{x}_i^{best}$, will end up being on $\mathbf{l}_i$ while the others will eventually be ignored. As the total energy of Eq. 5.3 is minimized, the $Obs^{silh}(\mathbf{x}_i^j, \mathbf{S})$ will collectively decrease in the least-squares sense and x_i^{best} will become closer and closer to actually being on the surface. Note that, because $\mathbf{x}_i^{best}$ minimizes the distance to the surface along the corresponding line of sight, the normal to the closest surface point is perpendicular to it. Thus, $\mathbf{x}_i^{best}$ will eventually tend to satisfy the two conditions that characterize a point on an occluding contour introduced in Section 6.1.2.

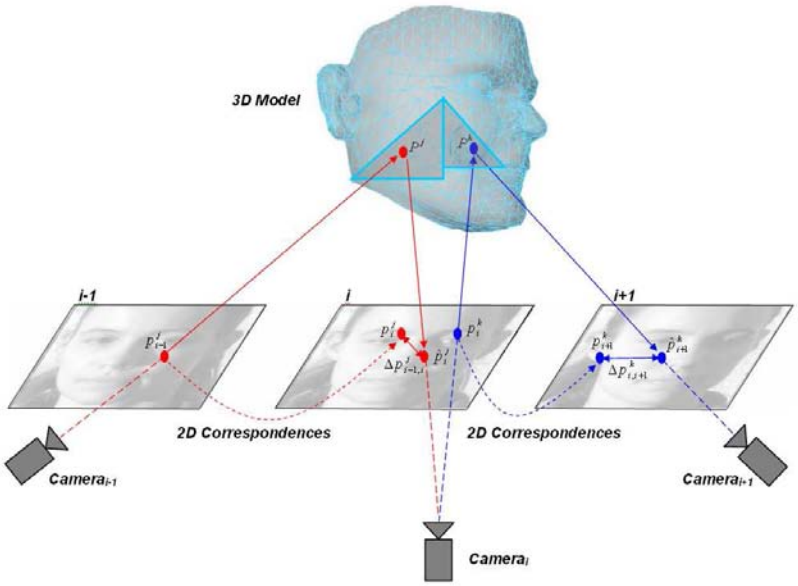


Figure 6.4: Back-projection procedure: First, the face area in the reference image, labeled as the image i in the figure, is densely sampled. Then correlation based algorithm is used to find corresponding feature points in one subsequent $i+1$ and one preceding $i-1$ image. The sampled points from the reference image are back-projected to the 3–D model, and intersection points on model's facets are found. These points are further projected to the side images. The sum of the squares of the distances between these back-projections and the corresponding points is minimized in terms of model parameters and camera parameters.

6.2.2 Correspondences

We use corresponding feature points in two similar contexts: one is for shape recovery of the deformable objects where the camera is considered static, and the other is for shape and camera motion recovery where camera is presumably moving around the subject. In the first context, corresponding feature points, has been used for tracking of deformable objects together with silhouettes. In the second context besides shape recovery, we also recover the camera motion. An example of tracking deformable object is presented in the Chapter 7, where the deformation of the moving piece of paper has been tracked. PCA based facial models are used for shape and camera recovery in wider context of bundle-adjustment. Both approaches use the notion of transfer function and the back-projection procedure.

First we will explain those concepts in case of simple shape recovery, and then we will discuss the bundle-adjustment context of shape and camera motion recovery.

6.2.2.1 Transfer Function in a Single Image Pair

We use 2–D point correspondences in pairs of consecutive images, as shown in Fig. 6.4 for pair of images i and $i+1$. We find interest points in the first image of the pair and establish correspondences in the second using a simple correlation-based algorithm. Given a couple $\mathbf{u}_i = (p_i^k, p_{i+1}^k)$ of corresponding points found in this manner, where k denotes feature point index of the i-th image frame, we define a *transfer function* $Obs^{trans}(\mathbf{u}_i, \mathbf{S})$ as follows: We back-project p_i^k from the i-th image to the 3–D surface and reproject it to $\hat{p}_{i+1}^k$the second image $i+1$. We then take $Obs^{trans}(\mathbf{u}_i, \mathbf{S})$ to be the Euclidean distance in the image plane between this reprojection $\hat{p}_{i+1}^k$ and p_i^k of the image i for all corresponding points $Q_i = \left\{p_i^k, 0 \leq k \leq N\right\}$:

$$Obs^{trans}(\mathbf{u}_i, \mathbf{S}) = \sum_{k \in Q} \left\| \hat{p}_{i+1}^k - p_i^k \right\| = \sum_{k \in Q} \left\| \triangle p_{i,i+1}^k \right\| \tag{6.5}$$

Note that the simplest and fastest way of backprojecting p_i^k to the surface is to use OpenGl and the graphics hardware of our machines to find the facet that is traversed by the line of sight defined by p_i^k. Therefore in our implementation, when computing $Obs^{corr}(\mathbf{u}_i, \mathbf{S})$ and its derivatives, we use the explicit representation instead of the implicit one.

For the tracking of deformable objects we repeat this procedure for each consecutive pair of images. However, when we also recover camera motion we can work with n images at the time, that will be explained bellow.

6.2.2.2 Transfer Function in a Sequence of Images

We use the sequence of images for facial reconstruction and camera calibration which are done simultaneously. For that purpose to the state vector $\mathbf{S}$, defining the model's shape, can be added vectors $\mathbf{C}_i$ containing ith camera position and orientation. The approach outlined above extends naturally to triplets of images $i-1, i, i+1$ in the sequence, given an an approximate value for $\mathbf{C}_i$. We create Q_i set of samples in image i, compute correspondences in the other two, and form the three-image objective function:

$$Obs_3^{trans}(\mathbf{S}, \mathbf{C}_{i-1}, \mathbf{C}_i, \mathbf{C}_{i+1}) = \sum_{j \in Q_i} \left\| \triangle p_{i,i-1}^j \right\|^2 + \left\| \triangle p_{i,i+1}^j \right\|^2 \tag{6.6}$$

In [34], we argued that minimizing Obs_3^{trans} is a well conditioned least-squares problem if we use enough correspondences and can therefore be used to derive reliable estimates of both camera and shape parameters. We can further refine this estimate by using additional images: We sample *independently* images $i-1$ and $i+1$ to create sample sets Q_{i-1} and Q_{i+1}, compute correspondences in images $i-2$ and $i+2$ and form the objective function:

$$\begin{aligned} Obs_5^{trans}(\mathbf{S}, \mathbf{C}_{i-2}, \mathbf{C}_{i-1}, \mathbf{C}_i, \mathbf{C}_{i+1}, \mathbf{C}_{i+2}) &= Obs_3^{trans}(\mathbf{S}, \mathbf{C}_{i-1}, \mathbf{C}_i, \mathbf{C}_{i+1}) \\ &+ \sum_{j \in Q_{i-1}} \left\| \triangle p_{i-1,i-2}^j \right\|^2 + \sum_{j \in Q_{i+1}} \left\| \triangle p_{i+1,i+2}^j \right\|^2 \end{aligned} \tag{6.7}$$

that we minimize with respect to all the parameters. This process can then be repeated recursively for the whole sequence and the objective function we end up minimizing is:

$$\begin{aligned} Obs_N^{trans}(\mathbf{S}, \mathbf{C}_1, ..., \mathbf{C}_N) = & \sum_{i=i_o}^{N-1} \sum_{j \in Q_i} \left\| \triangle p_{i,i+1}^j \right\|^2 \\ + & \sum_{i=2}^{i0} \sum_{j \in Q_i} \left\| \triangle p_{i,i-1}^j \right\|^2, \end{aligned} \tag{6.8}$$

where i_o is the index of the one image for which we need an initial pose estimate. Note, that because $\mathbf{C}_{i_o}$ is optimized at the same time as the other extrinsic camera parameters, that estimate need not be exact. This is made possible by the fact that in our approach, there is never an explicit association between 2–D sample points in the images and specific vertices or facets of the 3–D models. Instead, these associations are computed dynamically during the minimization and can change. In practice, we developed an optimization schedule in which the number of shape parameters that are allowed to vary progressively increases.

6.3 Summary

In this work we have presented a framework for the efficient detection and use of silhouettes for recovering the shape of deformable 3–D objects in monocular sequences. We rely on an implicit surface formalism that lets us look for occluding contours as solutions of an ordinary differential equation and to enforce the resulting constraints in a consistent manner.

To demonstrate the range of applicability of our method, we applied it to three very different problems: Reconstructing a PCA based face model from an uncalibrated video sequence; tracking a deforming piece of paper undergoing a partial occlusion or with a changing background; recovering head and shoulder motion in a cluttered scene.

In other words, our implicit surface based approach to using silhouettes is appropriate for uncontrolled environments that may involve occlusions and changing or cluttered backgrounds, which limit the applicability of most other silhouette-based methods. Furthermore, our approach is independent from the way the surface deformations are parametrized, as long as this parameterization remains differentiable.

7 Results

In this chapter we will demonstrate the possibilities of modeling with implicit meshes. We addressed two important Computer Vision problems: 3-D reconstruction from the uncalibrated video sequence, and tracking of deformable objects in monocular scenes. As we already discussed, implicit meshes are designed to take advantage of as many as possible data sources coming from the raw video sequences. We actually fitted the generic models turned into implicit meshes to: 3–D stereo data points, 2–D silhouettes and 2–D corresponding feature points depending on the application. The generic models we used are parametrized in terms of DFFD control points, PCA coefficients and simply vertices of the generic model. We tested our method for various examples of reconstruction and tracking, that can be listed as follows:

- reconstruction of the human upper body, including – head, neck and shoulders – from uncalibrated video sequences
- reconstruction of the human ear from pair of stereo images under the structured light
- reconstruction of the high resolution human face based on PCA parameterization from uncalibrated video sequence
- tracking of the moving head and shoulders in monocular video sequence and
- tracking of the deformable piece of paper in monocular sequence

We will start with 3–D reconstruction. First reconstruction experiment will be done on the synthetic example. This will be followed by the upper body and the ear reconstruction parametrized in terms of DFFD. Face shape recovery from uncalibrated video under the PCA parameterization will be addressed afterwards. Finally we will analyze performance of our approach and compare it with standard fitting techniques, which involve ordinary explicit surface representations. Also, the reconstruction examples will compare quality of the results using explicit surfaces and implicit meshes.

Second part will be devoted to the tracking of nonrigid motion of deformable 3–D objects in monocular sequences. Using the results of shape recovery from the reconstruction part we will track motion of the rising shoulders with the head that is not moving and later we will track both the head and shoulders motion, where the head performs rigid motion and the shoulders nonrigid one. We also track the motion of the deformable piece of paper in case of changing backgrounds and under partial occlusions in order to demonstrate robustness of implicit meshes in detecting and exploiting silhouette information.

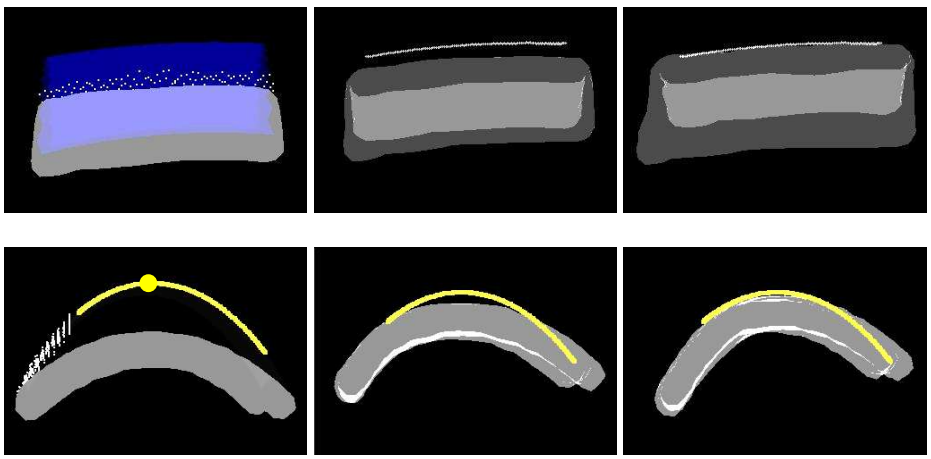


Figure 7.1: Synthetic example of fitting spherical implicit mesh. Left column: front and side view of the initial state–implicit mesh in light-grey, surface to fit in dark-gray whose presence is simulated by stereo data shown as white dots and silhouette shown as white line in front and thick white dot in side view. On the front view, we overlay the silhouette and, on the side view, we outline the top of the dark-gray object with a thick white line. Middle column: Fitting results using stereo alone. Right row: Fitting using both stereo and silhouette data demonstrates correct fitting.

7.1 3D Reconstruction

In our 3–D reconstruction examples we start from the generic models represented as arbitrary explicit triangulated surface, like once in Fig. 3.2. We then turn that surface into the implicit mesh as discussed in Section 4.1. Such representation is then used as generic model and should be deformed in such a way that it confirms to the data extracted from the images of the given video sequence. The video sequence we use are uncalibrated, with no a priori knowledge of camera parameters and its motion. We have first to register the images and recover camera motion, then to extract the data from images, such as 3–D stereo data cloud and 2–D silhouettes, to initialize our algorithm by positioning the generic model toward the data, and finally to optimize on the values of control parameters that deform the shape of the generic model.

7.1.1 Synthetic Example

We created a synthetic example that simulates a difficult situation in which one must combine stereo and silhouette data to achieve a good result. In this example, because the mesh is regular, we can use spherical or triangular primitives indifferently. Bellow we show results of fitting using first spherical and then triangular implicit mesh.

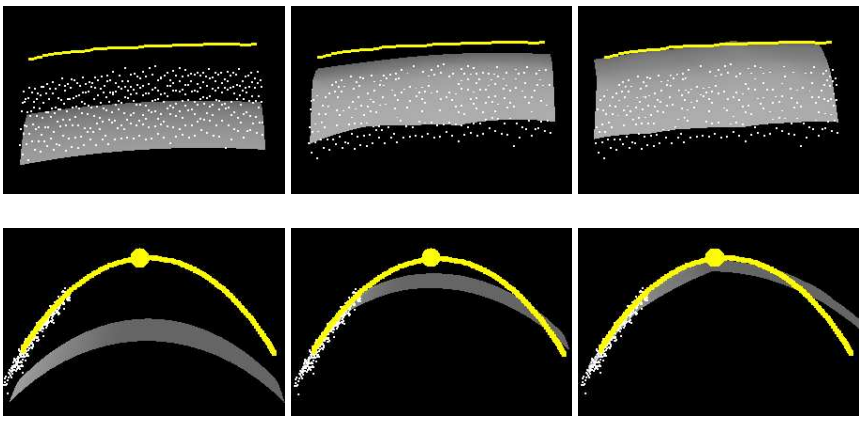


Figure 7.2: Synthetic example. Left column: Front and side view of an initial mesh shown in light-gray, with occluding contours of a cylindrical 3–D surface to be fitted drawn as white lines. The large white dot in the side view corresponds to the occluding contour in the front view. The smaller white dots represent simulated stereo-data. Middle column: Fitting results using stereo alone. Right column: Fitting results using both stereo and silhouette observations derived from the occluding contour in the front view. Note again the quality of fitting when silhouettes were used.

Spherical Implicit Mesh In the left column of Fig. 7.1, we show a side and a front view of a dark gray object that we want to fit using simulated 2–D and 3–D data. On the side view, we outline the top of dark-gray object with a thick white line. This object represents the surface we want to obtain after fitting. To make the problem realistic, we assume that we have stereo data, shown as white dots, only on the front side of the dark-grey patch, that is the one that faces the camera, and silhouette data at the top of it. This silhouette data is represented by the white line in the front image and by the white circle at the top of the dark-gray object in the side view. In these views, we also show, in light gray, an implicit mesh in its initial state. The middle column depicts the result of fitting using stereo alone. Note that, in the front view, the occluding contour of the deformed surface, shown in light grey, does not match the expected silhouette, again shown as a white line. Similarly, in the side view, it does not reach the white outline. The right column depicts the result using both stereo and silhouette data. The occluding contour of the fitted surface is now where it should be and the top of the surface has moved appropriately. The back of the shape is, of course, still inaccurate since there is neither silhouette nor stereo data to constrain it.

Triangular Implicit Mesh Fig. 7.2 depicts a synthetic example similar to one shown in Fig. 7.1. However, here we use triangular implicit mesh, and since it provides much better approximation of the explicit surfaces, we only show the explicit skeleton. The fitting is performed using triangular implicit surface around it. The left column of the figure depicts

Figure 7.3: Initial position of the generic model reprojected in the images.

the initial shape of an explicit mesh seen from the front and the side. Our goal is, again, to turn this explicit mesh into an implicit one and then to fit it to a surface whose outlines appear as white curved lines.

We assume that the cameras are in front of the surface we want to model and, therefore, yield stereo data, depicted by the white dots, only on the side facing camera and for the part of the surface that is close to being front-parallel. The middle column of Fig. 7.2 depicts the result of fitting using this stereo data alone. As could be expected, only the bottom part of the mesh is fitted correctly and the corresponding occluding contours do not match the white outlines. The right column of Fig. 7.2 depicts the result obtained by combining stereo and silhouette data. The top of the fitted mesh has moved appropriately and the occluding contours it produces are now where they should be. The back of the shape is again, still inaccurate since there is neither silhouette nor stereo data to constrain it.

7.1.2 Upper Body Reconstruction from Uncalibrated Video Sequence

We further demonstrate and evaluate our technique in the context of complete upper body modeling, that is including head, neck and shoulders, from uncalibrated video sequences. We use different image video sequences where we select only every fifth or tenth frame resulting in a small number of images from which we do the reconstruction. They were acquired with a video camera over a period of a few seconds where the camera is standing still and the subject is turning on the rotating chair or opposite. First we have to initialize our generic model, register the images, compute stereo and silhouettes and finally deform the model by fitting it to the extracted data. For the sake of comparison we repeat reconstruction process from the same video sequence using triangular implicit meshes, spherical implicit meshes and explicit meshes.

7.1.2.1 Initialization

We initialized the model by manually picking at least five 2–D points on the face in one reference image, that is one where the subject is facing the camera. We used them to compute a 4x4 rotation-translation matrix Rt such that five specific 3–D keypoints on the

generic model—outside corners of the eyes, corners of the mouth and tip of the nose—once multiplied by this matrix project as close as possible to the hand-picked 2–D location. Because these points are not coplanar, this guarantees that, when we multiply the generic model by this Rt matrix, we obtain an initial upper body model that is roughly aligned in the reference image.

In spite of relatively good alignment on the face as shown in Fig. 7.3, the shoulders are significantly far from the image silhouettes. However, our method showed insensitivity to this problem.

7.1.2.2 Image Registration

After the generic upper body model has been initialized we have to register the images of the provided video sequence. In the example of Fig. 7.4 we use as input an initially uncalibrated video sequence in which the subject is moving on the rotating chair in front of the fixed camera. We selected seven frames out of thirty frames long video sequence. Our generic model is initially aligned in the reference image. To register the images we used a model-driven bundle-adjustment technique of [47]. In short the bundle-adjustment algorithm goes through the following steps:

- Generate an initial 3x4 camera projection matrix Prj for the reference image, such that its principal point is in the center of the image. For the focal length we chose some approximate value. The intrinsic camera parameters are chosen some approximate values, while the camera position and orientation are not known.
- Compute initial camera position and orientation according to the already computed Rt matrix used to initialize the model. The initial projection matrix that refers to the reference image is taken to be $Tr = Prj * Rt$. This is identical to moving the camera such that it looks at the model where its keypoints project in the reference image close to the selected 2–D points.
- We further densely sample the facial part of the generic model and project those samples into the reference image and in one preceding and one subsequent image. We also match those points projected in the reference image to the preceding and subsequent image. We use simple correlation based algorithm of [48] to find matched points. We take the initial position and orientation of the two side cameras to be equal to the one in the middle, i.e. the reference one. The observation equations simply minimize the difference between the estimated (matched) points and measured (projected) points in respect to the camera position and orientation.

This yields the camera position for the two images on either side of the reference image. To compute the following camera positions, the image immediately succeeding the reference image becomes the new reference image. The procedure than repeats in triples of images as already explained. The same is done for preceding images, and continues until the end of the sequence.

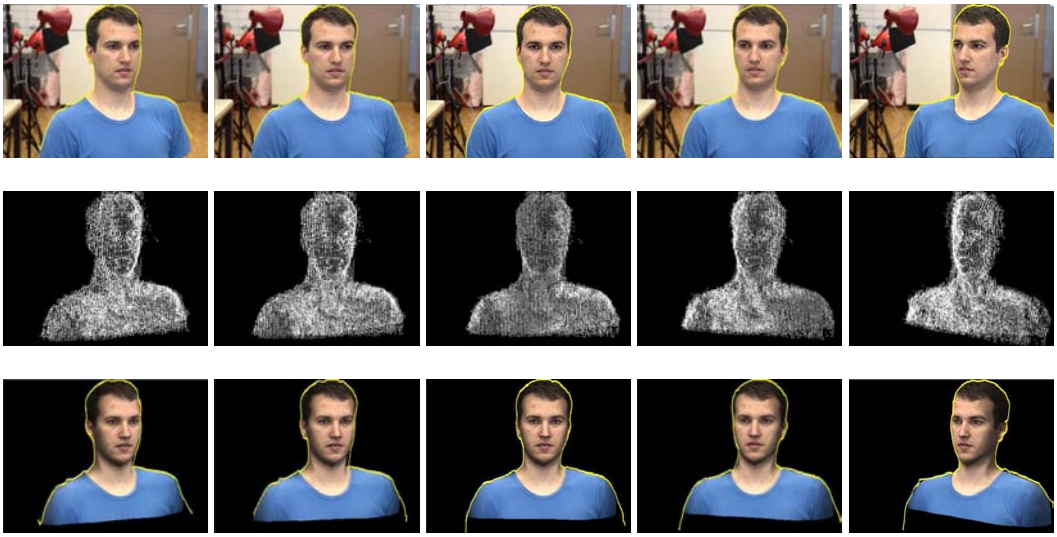


Figure 7.4: Reconstruction from a shorter uncalibrated video sequence. Top row: 5 of 7 images from a short video sequence with overlaid silhouettes on the head, neck and shoulders. Middle row: Clouds of 3–D points extracted from consecutive image pairs using correlation-based stereo, after automated registration. Bottom row: Textured reconstructed model with triangular implicit mesh model viewed in the same perspective as the original images and with overlaid silhouettes to highlight the quality of the fit.

7.1.2.3 Stereo and Silhouettes Computation

Having the whole image sequence calibrated we can easily compute stereo, that produces 3–D data points as ones shown in Fig. 7.4. The data cloud produced in this way is very noisy, because of assumptions made on the initial camera parameters, and also because of the missed matched points resulting from the correlation based technique. In some examples we used max-flow graph-cut algorithm [96] to derive disparity maps from consecutive image and produce the clouds of 3–D points as shown in Fig. 7.6. In this example silhouettes are computed using snakes starting from the manual outlines.

7.1.2.4 Reconstruction

Here we show reconstruction results, using real stereo and silhouette data. In the example of Fig. 7.4 we use as input an initially uncalibrated 7–frame video sequence in which the camera is static and subject is moving on the rotating chair. We first used a model-driven bundle-adjustment technique [47], as explained above, to compute the relative motion and, thus, register the images. We then derived the stereo data depicted by the middle row of Fig. 7.4 using a correlation-based stereo technique [46]. We used snakes to outline the silhouettes shown as white lines.

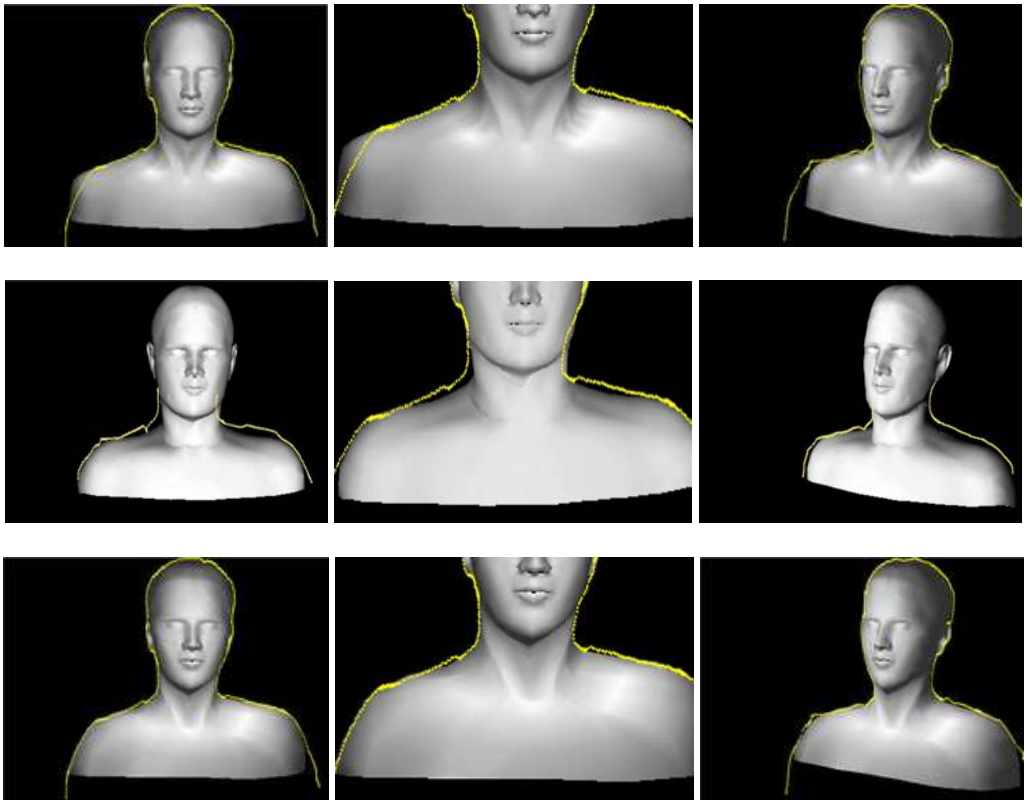


Figure 7.5: Comparing explicit and implicit approaches to fitting an upper body model to the stereo and silhouette data of Fig. 7.4. Top row: Directly using explicit surfaces yields a poor fit on the shoulders and the right side of the face, as evidenced by the discrepancies between the surfaces' occluding contours and the true ones shown as white lines. Bottom row: Using triangular primitives results in a much better correspondence.

The results of Fig. 7.4 were obtained using triangular implicit mesh. For comparison purposes, in Fig. 7.5, we show the result of fitting the model to the same data using directly the explicit surface without taking advantage of the implicit surface formalism proposed in this paper. In all cases, the explicit mesh is parametrized in terms of the same DFFD control points depicted by Fig. 3.2(b). When *not* using the implicit surfaces, we minimize the objective function of Section 5.3.2, which unlike those of Section 5.3.1 is non-differentiable and highly sensitive to the regularity of the mesh facets especially when it comes to handling silhouette information. This results in the fit depicted by the top row of Fig. 7.5 that is inaccurate in the shoulder area, which is where the silhouettes are the primary source of information.

The spherical primitive result shown in the second row is better, but still imprecise at

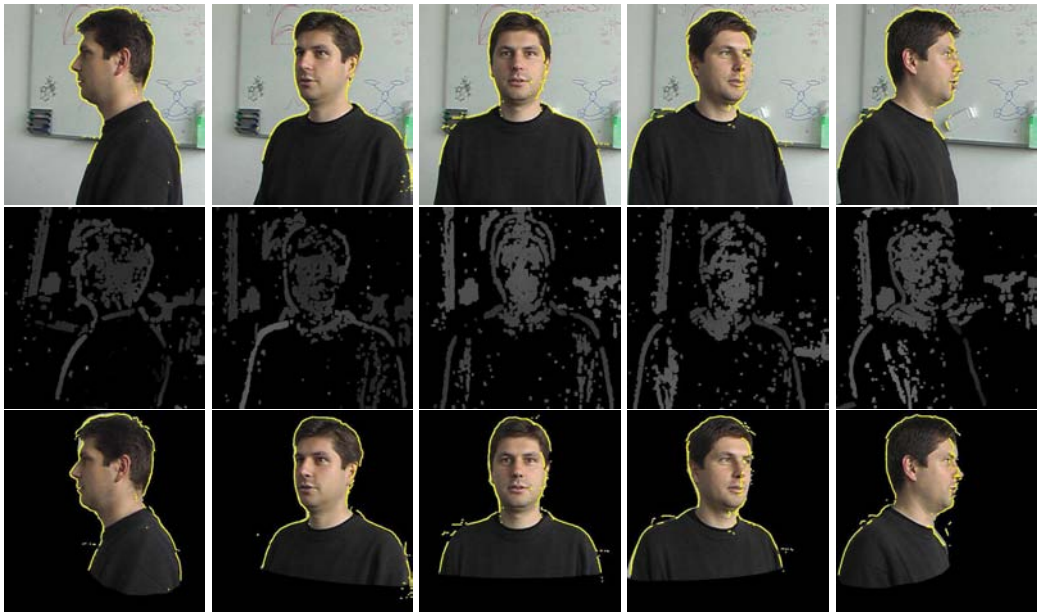


Figure 7.6: Reconstruction from an uncalibrated video sequence. Top row: 5 of 14 images from a short video sequence with overlaid silhouettes for the neck and shoulders. Middle row: Disparity maps computed from the image using a correlation-based stereo algorithm, after automated registration. Bottom row: Textured reconstructed models obtained by using a triangular implicit mesh model for the upper body.

the junction of the neck and shoulders. Close examination of the results show that this is a manifestation of the problem discussed in Section 4.1.1 and depicted by Fig. 4.5: Because the facets of the neck and shoulder are of different sizes, the thickness of the implicit surface varies and, at places, the wrong side of it ends up being attracted to the data.

As shown in the bottom row of Fig. 7.5, these problems go away when the spherical primitives are replaced by triangular ones. Note that this is true, even though we fitted the irregular low resolution mesh from the bottom row of Fig. 4.4 instead of the high resolution one from the top row of Fig. 4.4 that we used both to directly fit the explicit surface and in conjunction with the spherical metaballs.

Thus, the bottom row of the Fig. 7.4 depicts the reconstructed and textured model, obtained using triangular implicit meshes and whose projections align correctly with the silhouettes in all views. This shows that the recovered shape is geometrically correct even at places where the surface slants away from the cameras and, therefore, where stereo fails. Note that these texture-mapped views were generated using standard OpenGL tools to render the explicit surface that was deformed in tandem with the implicit one. In other words, having both kinds of representation simultaneously available at the same time spared us the need to use sophisticated implicit surface rendering techniques.

In another example of Fig. 7.6 we show a similar behavior. Now we use as input an initially uncalibrated 14–frame video sequence in which the subject moves in front of a static camera. We again ran a correlation-based stereo algorithm [46] to derive disparity maps from consecutive image, depicted by the middle row of the figure, and produce the clouds of 3–D points. Our automated silhouette detection was used to outline the silhouettes shown as white lines. The bottom row of the figure depicts the reconstructed and textured model, obtained using triangular implicit meshes and whose projections align correctly with the silhouettes in all views. Note that besides some noise in silhouettes we obtained accurate reconstruction.

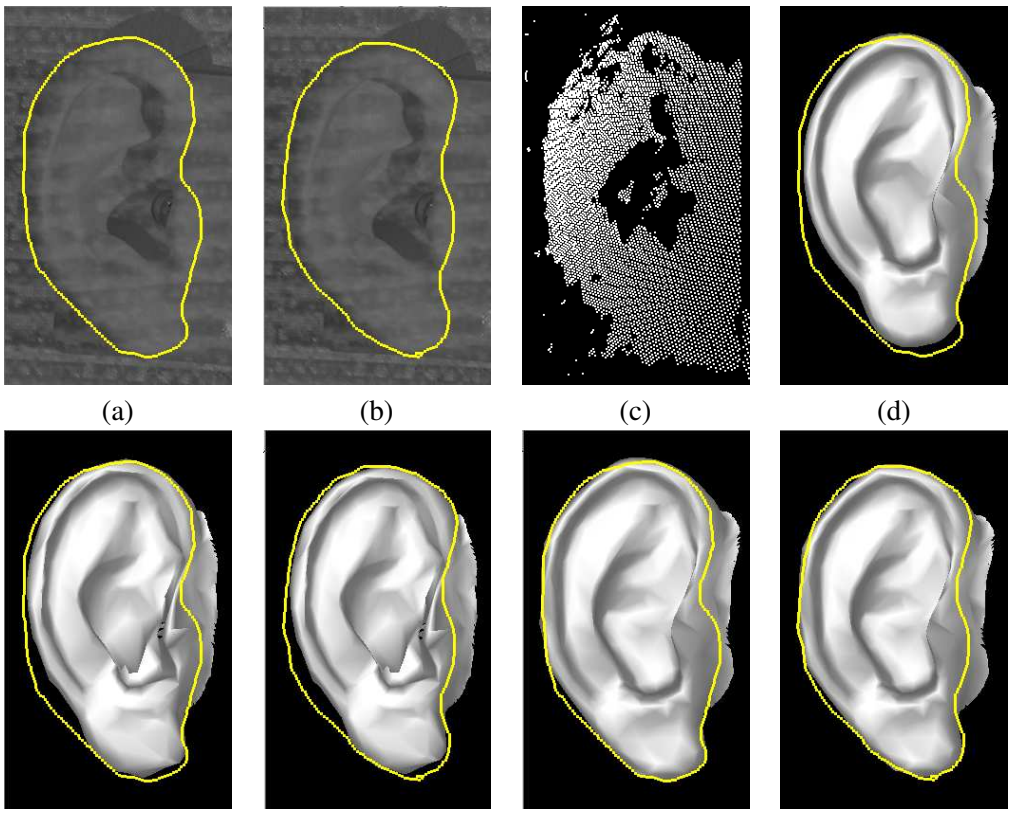


Figure 7.7: Modeling an ear. (a,b) A stereo pair with overlaid occluding contours. (c) The corresponding cloud of 3D points. (d) Projection of the initial ear model into one of the images. (e,f) Projections into both images of the model fitted using explicit surface. (g.h) Similar projections for the model using triangular primitives.

7.1.3 Ear Shape Recovery

In our next example, we consider a human ear, whose shape is more complex than the comparatively simple upper body models we have used so far. The challenge also comes from the fact that the model is taken from the web, and is not designed with fitting in mind. It is heavily irregular and has variety of different sized facets.

As shown in the top row of Fig. 7.7, we projected textured light on an ear and acquired a stereo pair of images, which allowed us to compute a fairly dense disparity map out of which we computed stereo data cloud depicted in Fig. 7.7(c). We then outlined occluding contours in both images and fitted a model we found on the web to this stereo and silhouette data, using the explicit mesh and triangular implicit mesh. Spherical metaballs were not possible to use in this case without additional re-meshing of the model. Re-meshing means increase in the mesh resolution and regularization of the facets.

In Fig. 7.7(d) we projected the initial model into the image in order to depict the gap to be filled between the model and the silhouette outlines. Result of fitting explicit mesh model is depicted in the Fig. 7.7(e,f) and, obviously, is not correctly aligned with the silhouettes. This comes from the fact, that silhouette facets are hard to determine on the thin border of the ear surface. Again, only the results obtained using triangular primitives and shown in the bottom row of Fig. 7.7(g, h) correctly line up with the silhouettes and appear realistic. Note the importance of the silhouette information, since even with the relatively high quality stereo data obtained from structure light, the result of fitting just to stereo could not be sufficient.

7.1.4 PCA Face Shape Recovery from Uncalibrated Video Sequence

In this section, we show results of our approach to head modeling from uncalibrated image sequences. We use 2–D point correspondences in pairs of consecutive images as our main source of information because the disruptive effect of illumination changes are minimized for images whose viewpoints are close and can be further attenuated by normalization. The results obtained in this way are correct everywhere except on the places where the surface slants away from the camera, actually on the occluding contours. For that reason we included the silhouette information in a way explained in Chapter 6 including: automatic silhouette detection in the images, and fitting the face model to those silhouettes in order to correct reconstruction obtained using only correspondences.

In theory, given one image for which the 3–D head pose is roughly known and a second one seen from a relatively similar viewpoint, we could recover shape and camera position as follows:

- Sample the face area in the first image.
- Use a straightforward normalized cross correlation [44] technique to find corresponding points in the second image.

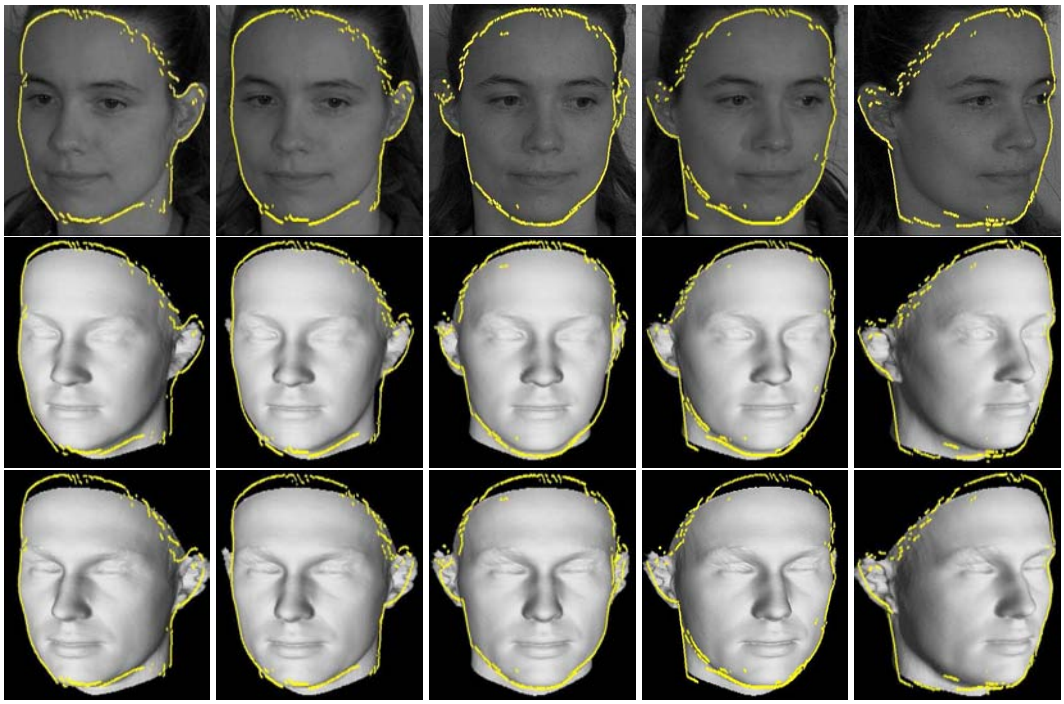


Figure 7.8: Head modeling using PCA face models. Top row: Five images from a short video sequence with image silhouette edges detected by using our technique. Middle row: Recovered face shape using only interest points with the same silhouettes as before. Note that they do not match exactly. Bottom row: Recovered shape using both silhouettes and interest points. The occluding contours of the model now corresponds almost exactly to the silhouette edges.

- Minimize in the least-squares sense the image distance between these corresponding points and those obtained by backprojecting the points in the first image to the model and reprojecting them into the second image.
- Detect silhouettes in the images and minimize the distance between model and detected silhouette points using implicit meshes together with the interest points

In practice, because correspondences can be expected to be noisy, we use an iterative reweighted least square technique and, more importantly, we work with more than two images simultaneously. Our complete approach therefore iteratively adds images at both ends of the sequence by establishing correspondences between the first and last images that have already been processed and neighboring ones that have not been considered yet. At each step of this process, we perform a least-squares minimization that progressively refines model shape and pose for *all* cameras. Unlike earlier approaches [102], our method has no notion of a *reference* image and the pose in the first image does not need to be pre-

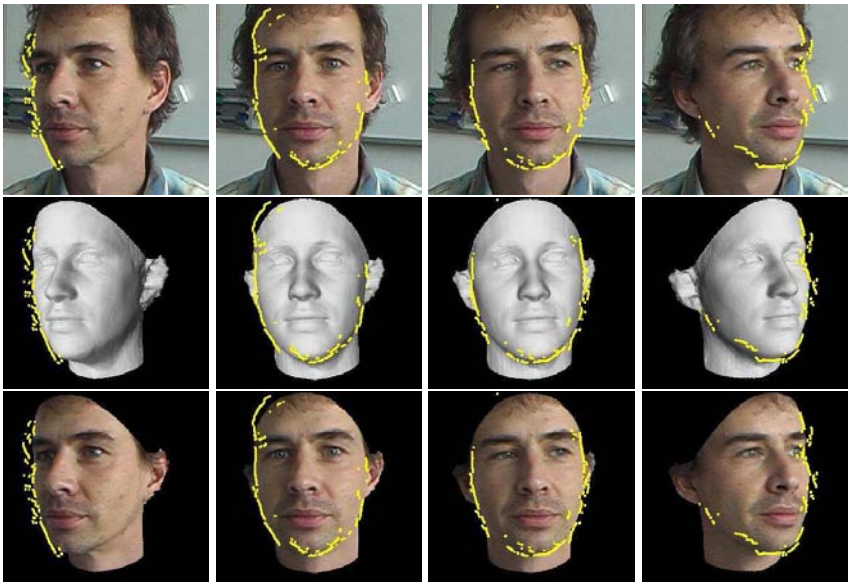
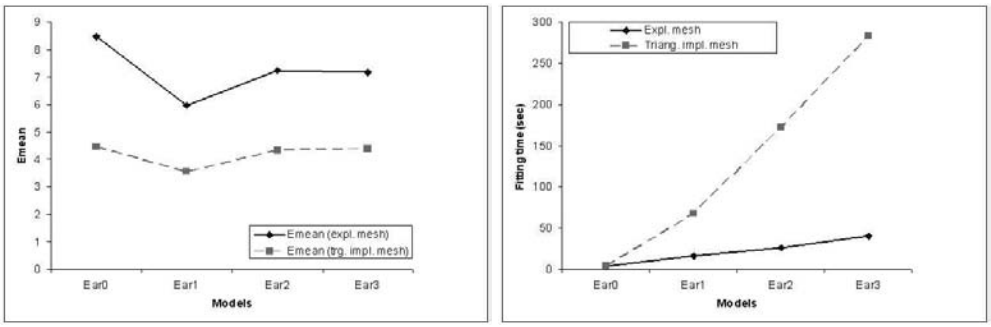


Figure 7.9: Head modeling using PCA face models. Top row: Four images taken out of the short video sequence . Middle and bottom row: Reconstructed face, viewed from different recovered camera views, is shown as shaded and textured model obtained by fitting matched interest points and superposed silhouettes. Note that the model does align correctly the extracted silhouettes.

cise because it will be refined as all the others. Once the sequence is calibrated, we can detect silhouettes in the images. To do this we convert the explicit face mesh model into our implicit mesh and then find the occluding contour on the model for each camera position. Than we correct model's shape by fitting those silhouettes together with the interest points.

We used short video sequence from which we selected seven images. Five of them are depicted in the top row of Fig. 7.8 together with automatically extracted silhouettes obtained using implicit mesh models. Recovered face shape using only interest points with the real silhouettes detected in the images is depicted in the middle row of Fig. 7.8. Note, that the resulting shape is not correctly aligned with the real silhouette edges. Finally, in the bottom row of Fig. 7.8 the recovered shape obtained when we added silhouette observations corrected the miss-alignment appearing when only corresponding feature points were used for shape recovery.

In Fig. 7.9 we depict another example of face shape recovery from the short video. The lightening conditions are quite bad, and our silhouette detection algorithm did the best to extract correct occluding edges. Note that the obtained result align correctly the extracted silhouettes.



(a) (b)

Figure 7.10: Measured fitting errors and computation times as a function of increasing resolution. (a) Mean distance in millimeters of explicit and implicit mesh to the silhouette data. (b) Computation times for explicit and implicit mesh fitting on a 2.6GHz PC in seconds.

7.1.5 Accuracy and Computational Complexity

To quantify fitting error and computational complexity, we started with Ear_0(1167 vertices, 1620 facets), the model of Fig. 3.2(c). We then subdivided its facets to obtain three additional models of increasing complexity: Ear_1(3954 vertices, 6480 facets), Ear_2(14388 vertices, 25920 facets) and Ear_3(54696 vertices, 103680 facets). Fig. 7.10(a) depicts the mean distance of the lines of sight defined by the silhouette points to the models fitted with or without using implicit meshes. When using them, the accuracy varies little with mesh resolution and, as observed before, is much better than the one obtained without them. As shown in Fig. 7.10(b), there is a computational price to be paid for using implicit meshes. Note however that, because the result is fairly insensitive to mesh resolution, there is no advantage to subdividing the mesh and one need therefore not incur this penalty. Furthermore, even for the model with 103680 facets, the computation time remains manageable on a modern machine.

To gauge the influence of the regularization constant λ_D of Eq. 3.9, in Fig. 7.10(c), we plot the value of $\sqrt{(\sum w_{type_i} \epsilon_i^2 / \sum w_{type_i})}$, the root mean square residual of our fit using the notations of Eq. 5.3. For very small values of λ_D, the influence of the regularization term is insufficient to "convexify" the problem and the optimizer tends to get trapped in meaningless local minima. For very large values of λ_D, the model becomes too stiff again resulting again in increased errors. However, between these extremes, there is a large range in which the result is fairly insensitive to the exact value of λ_D, making it easy to pick an appropriate value.

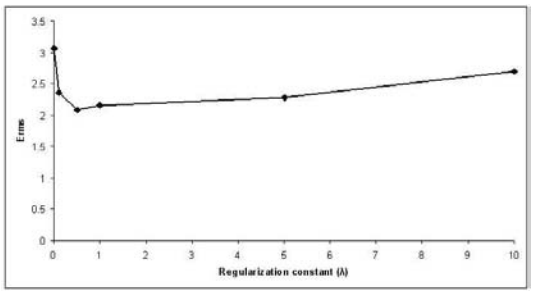


Figure 7.11: Influence of the regularization constant λ on the root mean square error of the fit.

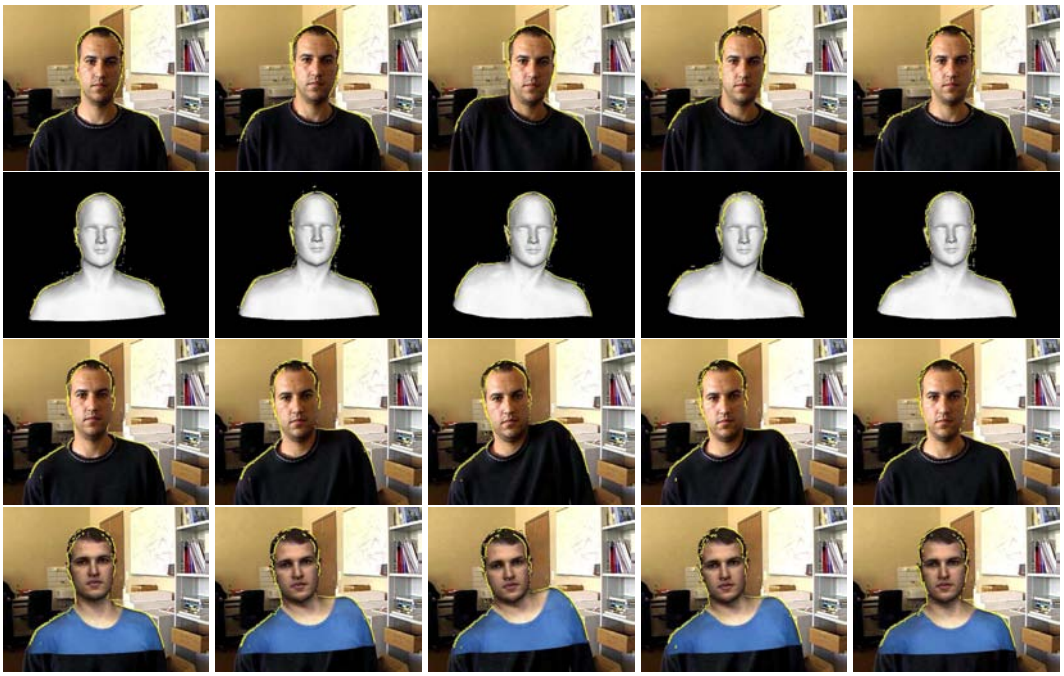


Figure 7.12: Tracking of the moving head and shoulders. Top and third row: Images from the original video sequence with detected silhouette edges overlaid. Second row: Recovered model's shape shown as the shaded model in the same position as the original images above. Bottom row: Recovered model's shape shown as the textured model in the same position as the original images above. Note that silhouette edges are correctly fitted.

7.2 Monocular Tracking

This section will be devoted to the tracking of nonrigid motion of deformable 3–D objects in monocular sequences. Using the results of shape recovery from the reconstruction part

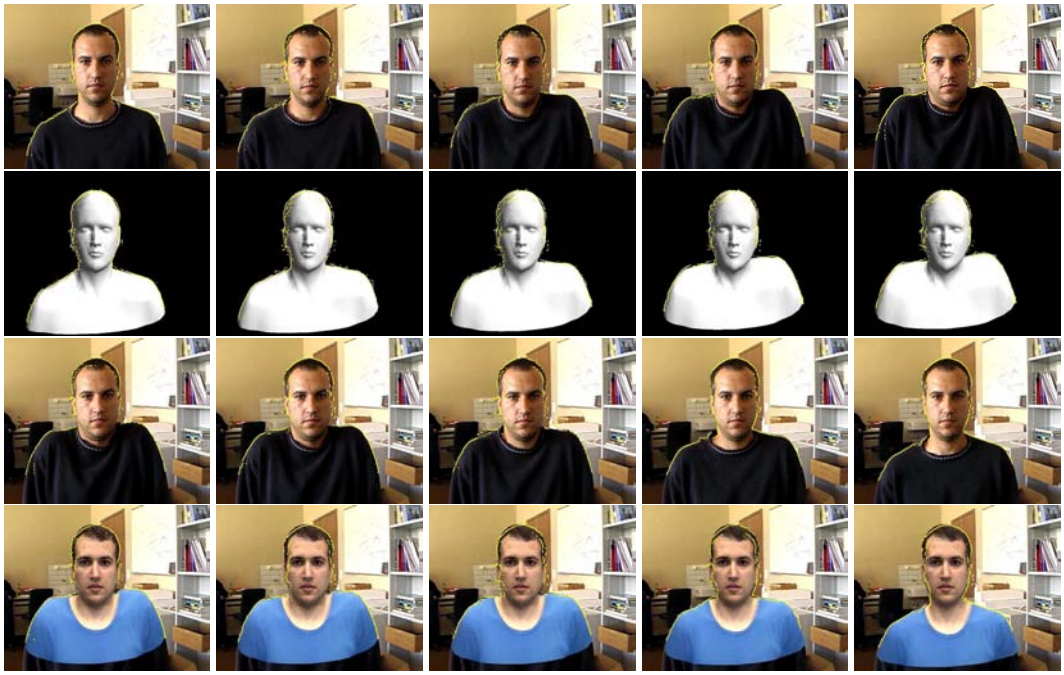


Figure 7.13: Tracking of the rising shoulders. Top and third row: Images from the original video sequence with detected silhouette edges overlaid. Second row: Recovered model's shape shown as the shaded model in the same position as the original images above. Bottom row: Recovered model's shape shown as the textured model in the same position as the original images above. Note that silhouette edges are correctly fitted.

we track the motion of the rising shoulders with static head and later we track simultaneous the head and shoulders motion in video with very cluttered background. We also track the motion of the deformable piece of paper in the cluttered and changing background and under partial occlusions in order to demonstrate robustness of implicit meshes in detecting and exploiting silhouette information.

7.2.1 Head and Shoulders Tracking

We shoot two video sequences of the person, which we already reconstructed as shown in Fig. 7.4. New sequences are shot with different camera, and the person is wearing different clothing then it was the case in the sequence used for reconstruction. The cameras were fixed while the subject was moving. Here we apply our method to recovering the motion of moving head and shoulders in monocular sequences, where the model is parameterized in terms of DFFD control points.

We first build a 3–D model from a sequence where the camera does not move but the

person does, as it is explained above and is shown in Fig. 7.4. This model is then used for tracking in sequences such as the one in the Fig. 7.13 and Fig. 7.12. We initialize by positioning previously reconstructed model according to the five points on the face. We then, compute the occluding contour of the model that is used, and project it into the image. Detected real silhouette edge is then fitted in order to correctly align the model's outline with the silhouette in the image. This procedure is subsequently repeated throughout the sequence: silhouette of the model in time $t-1$ is projected into the image in time t and the distance between the model's occluding contour and the real silhouette is minimized. In the Fig. 7.13 we tracked just rising of the shoulders, where the head was not moving. For this example we additionally, fixed all the control parameters controlling the shape of the head, and optimized only on the neck and shoulders control points positions. In this case tracking is only based on silhouettes. In Fig. 7.13 the first and third row depict image frames from the original video sequence together with detected silhouette edges. In the second row we show shaded model with overlaid the same detected silhouettes as depicted in original images. Note that the reconstructed model correctly aligns the shoulders. In the bottom row we showed the textured model again in the positions corresponding to the images above. Since the model used for reconstruction comes from different video sequence, thus the texture is not the one of the subject from the tracking sequence. Note that in this way we augmented the scene by showing the same person but in different clothing.

In order to fully take advantage of our implicit mesh formalism we investigated another case where the head and shoulders were simultaneously moving. In this case, interest points are found on the head while occluding contours are used for the neck and shoulders. This results in the reconstruction of Fig. 7.12. In this case rigid head motion is retrieved using interest points, while nonrigid motion of the neck and shoulders is obtained according to the silhouette information. Again, we showed, in first and third row images from the original video sequence with detected silhouettes contours and bellow reconstructed model shown as shaded and textured again with overlaid silhouettes in order to demonstrate tracking precision.

As shown in Fig. 7.14, the reconstructed model can be used to resynthetize the subject in front of a different background, thus eliminating the need for a blue screen. This, once again, highlights the robustness of the silhouette detection guided by our implicit meshes.

7.2.2 Tracking a Deformable Piece of Paper

We model the paper as a rectangular mesh parametrized in terms of the coordinates of its vertices. The piece of paper is put on the flat surface and is taped on the end closer to the camera, while it is pushed forward from the other end as it is shown in Figs 7.15, 7.16. There is one fixed camera filming the paper's nonrigid motion. In the first frame, where the paper is flat and is in its still position, we select four corners of the paper for which we know corresponding models 3-D points. Considering the model as a plane $z = 0$ we can establish homography between that plane and the image plane, which will give us the initial camera position and orientation. We suppose to know the camera's internal parameters. In this way we initialized the model, so that upper left corner of the paper is used as the origin of the

Figure 7.14: Tracking of moving head and shoulders. The model was first reconstructed from an uncalibrated video sequence of Fig. 7.4 from which the texture was taken. Top row: Original sequence used to track the person. Bottom row: Recovered model placed in front of a different background.

global coordinate system, and the x and y axis are aligned with two sides of the paper.

We track paper's motion according to the corresponding feature points established between consecutive image pairs and already explained transfer function and the backprojection of Section 6.2.2. Also, we use online computed occluding contours using our implicit mesh formulation for robust silhouette detection. Besides occluding contours we fit our model to the border edges. This helps in additional constraining of the model's position towards the edges. For that reason we write observation equation for border edges as follows:

$$Obs^{bord}(\mathbf{x}_j, \mathbf{S}) = \sum_{\mathbf{x}_{j \in D}} \|\mathbf{x}_j - edge(\mathbf{x}_j, \mathbf{S})\| \tag{7.1}$$

where $\mathbf{x}_j$ is the border edge point in the image from the set D of available samples and $edge(\mathbf{x}_j, \mathbf{S})$ is the sample border edge point of the given model.

Fig. 7.15 shows the results obtained when the paper is partially occluded. The top and third row show the deformed mesh we obtain overlaid as a white wireframe on the original images. The second and bottom row show the side view of the same deformed mesh. We can see that the back of the mesh also deforms in a coherent manner. Even though the silhouette contours are partially hidden, our algorithm still retrieves the correct deformation and keeps on tracking the piece of paper.

Fig. 7.16 highlights the robustness of our algorithm to a changing background. The top and third row show the original sequence with the same tiger image as before and a moving book behind. In the second and bottom row, we used the deformed mesh to map a new texture onto the images. The new images look realistic and such results couldn't have been obtained by using a simple background substraction technique.

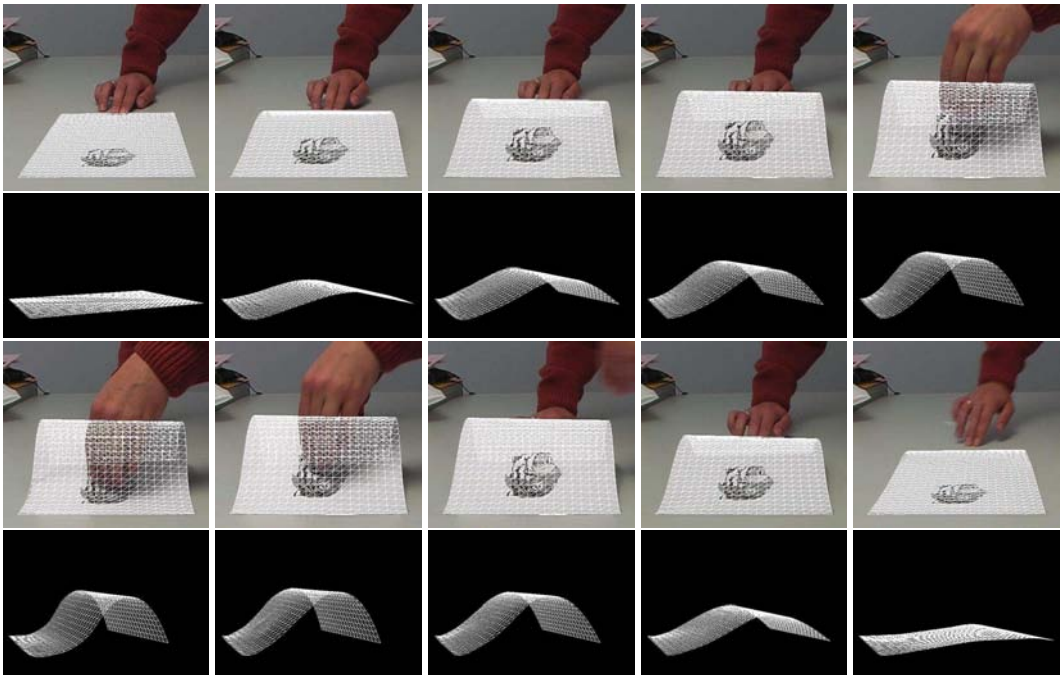


Figure 7.15: Occlusion handling. The front of the paper is taped to the table and one hand pushes the back of the page while the other passes in front. Top and third row: The recovered mesh is overlaid on the images. Note that the hand is *in front* of the paper even though the wireframed display gives the impression that it is behind. Second and bottom row: Side view of the recovered mesh. Note that its shape is undisturbed by the occlusion and that the back of the mesh also deforms correctly.

7.3 Summary

We used the example of upper-body and ear modeling using stereo and silhouette data to demonstrate the power of implicit mesh approach. The explicit models we used, and which were easily converted to implicit meshes, were not tailored for fitting purposes and exhibited both highly irregular facets and a complex topology, none of which had a significant impact on the quality of the fitting. However, with pure explicit surfaces, we encountered problems, especially when fitting them to silhouettes. This approach is effective independently of the specific way the deformations are parametrized. Reconstruction of a human face parametrized in terms of a Principal Component Analysis model [7, 34] is another example we have shown here.

We also tested our implicit meshes for tracking deformable object. We validated the tracker in two very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh and tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations [84]. Note the robustness

Figure 7.16: Handling a changing background. Top and third row: Original sequence with book sliding in the background. Second and bottom row: A new texture is applied on the deformed mesh and reprojected in the images. Note that background subtraction techniques could not have been applied in this case.

of silhouette handling in cases of cluttered and changing background, and in case of partial occlusion.

8 Conclusion

8.1 Contribution

We have presented an approach to combining explicit and implicit surface representations that allows us to take advantage of the strengths of both. To this end, we have developed a technique for creating *implicit meshes* from explicit ones by attaching triangular or spherical primitives to their facets. These primitives are defined in such a way that their shape depends only on the 3D location of the mesh vertices, which allows us to simultaneously fit both representations to image data by minimizing a differentiable objective function. Such surface representations allows efficient usage of various data sources coming from images. In our experiments we demonstrated effective use of this representation to fitting stereo, silhouettes and interest points observation data.

We also demonstrated the power of using DFFD shape deformation approach to parameterize deformable surface models and fit them to noisy 3–D image data. This resulted in a generic approach to surface parameterization since it can be applied to surface models of any geometry and complexity. In terms of fitting they offer significant dimensionality reduction by expressing the model surface with much smaller number of control parameters. We demonstrated the effectiveness and robustness of DFFDs in the context of complete head modeling from stereo data. We later used DFFDs to deform implicit surfaces as opposed to the explicit ones we used at the beginning. Combining both approaches therefore produced an even more powerful modeling tool.

The implicit mesh framework have been also used for the efficient detection and use of silhouettes for recovering the shape of deformable 3–D objects in monocular sequences. This again relays on an implicit mesh formalism that lets us look for occluding contours as solutions of an ordinary differential equation and to enforce the resulting constraints in a consistent manner. We have chosen to use DFFD control points and PCA parameters, to parameterize the position of the mesh vertices, which allows us to perform this minimization with respect to a limited number of parameters. Our implicit meshes, however, are generic and we could also have used another surface parameterization approach for this purpose.

To demonstrate the range of applicability of our method, we applied it to four different problems: upper-body and ear modeling using stereo and silhouette data where the initial generic explicit models we used were not tailored for fitting purposes and exhibited both highly irregular facets and a complex topology, none of which had a significant impact on the quality of the fitting; reconstructing a PCA based face model from an uncalibrated video sequence, where we besides shape recovery we also recover the camera motion; tracking a deforming piece of paper undergoing a partial occlusion or with a changing background; re-

covering head and shoulder motion in a cluttered scene. In other words, our implicit surface based approach to using silhouettes is appropriate for uncontrolled environments that may involve occlusions and changing or cluttered backgrounds, which limit the applicability of most other silhouette-based methods.

8.2 Extensions and Future Work

In this work we proposed algorithm which can convert arbitrary triangular mesh to the implicit surface. In that case we used spherical or triangular metaballs. It is possible to generalize this to meshes made of any kind of polygons. For that purpose, the generic polygonal metaball has to be defined. This would be particularly useful if someone wants to use NURBS or subdivision surfaces.

In the example of paper deformation, we already tried to model physically meaningful object deformation. The best way to deal with such physically based deformations is to use right model parameterization. Such parameterization will be based on the real physical properties of the materials whose nonrigid motion we want to recover. Usual way of solving this problems is writing the deformation energy in terms of the model topology, which is solved using Finite Element Methods (FEM). Implicit mesh can serve for handling silhouette information, that is indispensable source of information in this case. In this way we cloud still broaden the applicability of our model.

For tracking of deformable objects we did not consider potential global motion of the deformable objects. This can be very useful in some particular applications, such as tracking of the deformable sail, or shoulders tracking of the runners, where there is both global upper body motion and deformable motion of the shoulders. In this case we should use separate optimization procedure in two steps: one would retrieve global position in the current frame from the position of the previous frame, and then the local adaptation would be performed to obtain correct deformation.

More sophisticated edge search algorithms then the one which is based on the image gradient and the search in the normal direction has to be used. One of these methods is based on the texture boundary detection [101]. For those algorithms, it is sufficient to provide the points of the occluding contour and its normals. The algorithm should provide better silhouette edges on the boundaries where two textures differentiate.

9 Appendix

9.1 Differentiability of the Triangular Metaballs Field Function

Theorem 1. Distance function $d(\mathbf{x}, \mathbf{S})$ defining distance of the point $\mathbf{x} \in R^3$ to the triangle F_i, defined by vertices $\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3\}$ of the triangle whose position is controlled by the set of control parameters $\mathbf{S}$, is C^1 differentiable and continuous both over the parameters $\mathbf{S}$ and over independent variables x, y and z.

Proof: Considering separately distance functions representing distance of the point $\mathbf{x}$ to the plane d_p, to the line d_c and to the point d_s we can say that for every point which is not on the border of connecting regions those functions are C^2 differentiable both over the parameters and independent variables x, y and z. First order derivatives over parameters are:

$$\begin{aligned}
\frac{\partial d_p(\mathbf{x}, \mathbf{S})}{\partial s} &= \mathbf{x}^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \mathbf{x} + \mathbf{x}^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} \mathbf{x} \\
\frac{\partial d_c(\mathbf{x}, \mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_c \frac{\partial T_r(\mathbf{S})}{\partial s} \mathbf{x} \\
\frac{\partial d_s(\mathbf{x}, \mathbf{S})}{\partial s} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_s \frac{\partial T_r(\mathbf{S})}{\partial s} \mathbf{x}
\end{aligned}$$

while first order derivatives over independent variables x, y and z are:

$$\begin{aligned}
\frac{\partial d_p(\mathbf{x}, \mathbf{S})}{\partial x_i} &= \frac{\partial \mathbf{x}^T}{\partial x_i} L_1 L_2 \mathbf{x} + \mathbf{x}^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} = 2 L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i}, \\
&\quad \frac{\partial \mathbf{x}}{\partial x_i} = [x_1, x_2, x_3, 0]^T, x_i = 1, x_{k \neq i} = 0 \; and \; L_1 L_2 = (L_1 L_2)^T \\
\frac{\partial d_c(\mathbf{x}, \mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_c T_r \frac{\partial \mathbf{x}}{\partial x_i}, \frac{\partial \mathbf{x}}{\partial x_i} = [x_1, x_2, x_3, 0]^T, x_i = 1, x_{k \neq i} = 0 \\
\frac{\partial d_s(\mathbf{x}, \mathbf{S})}{\partial x_i} &= 2\mathbf{x}^T T_r^T(\mathbf{S}) L_s T_r \frac{\partial \mathbf{x}}{\partial x_i}, \frac{\partial \mathbf{x}}{\partial x_i} = [x_1, x_2, x_3, 0]^T, x_i = 1, x_{k \neq i} = 0
\end{aligned}$$

Consider now derivatives at the border lines where different regions, thus different distance functions are used. With no loss of generality proof will be done for:

1. points along half circles between regions $reg4$ and $reg7$ that is a border curve between cylinder aligned with x-axis and the sphere at the origin.

2. points which are on the line passing through vertex $\mathbf{P}_1$ in the direction of the normal $\mathbf{n}$, where four regions $reg1,\ reg4,\ reg7,\ reg3$ meet, and which can be represented as: $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}$

3. points along the lines parallel to the edge $\mathbf{P}_2\mathbf{P}_1 = \mathbf{m}_1$which separates regions $reg1$ and $reg4$, and can is given by: $\mathbf{x}_2 = \mathbf{x}_1 + \eta\mathbf{m}_1 = \mathbf{P}_1 + \lambda\mathbf{n} + \eta\mathbf{m}_1$

Since transformation matrix T_r can be created such that local coordinate frame is attached to either of the triangle vertices, and aligned with either of the edges we can consider the proof of upper three cases extendible to other border curves, points and lines.

CASE 1: This is a trivial case, since at the connection of the sphere and cylinder matrix defining a sphere L_s becomes equal L_c since we consider sphere in the plane $x = 0$. For clarity consider a simple case of standardized facet with vertices $\mathbf{P}_1 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T, \mathbf{P}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \mathbf{P}_3 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, where $Tr = I$. In this case distance function from the line along x-axis, in this case $\mathbf{P}_2\mathbf{P}_1$, becomes a cylinder equation $d_c(\mathbf{x}) = y^2 + z^2$, and distance from the point, in this case $\mathbf{P}_1$, becomes sphere equation $d_s(\mathbf{x}) = x^2 + y^2 + z^2$. For the border curve between this cylinder and the sphere we consider all the points on the circle $y^2 + z^2, y > 0$. In this case distance function from the sphere in the plane $x = 0$ becomes $d_s(\mathbf{x}) = y^2 + z^2$ that is actually identical to the cylinder distance d_c. Having the same functions along this curve in the plane $x = 0$ they certainly have the same first order derivatives for all points on this curve.

CASE 2: Let us consider all the points $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}$ on the connection of four regions $reg1,\ reg4,\ reg7,\ reg3$ where those regions actually meet. We first compute first order derivatives over parameters at these points for each of the four distance functions, and then first order derivatives over the independent variables x, y and z.

First order derivatives over the parameters in the points $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}$:

1. Derivatives of the distance to plane d_p function:

$$\begin{aligned}
\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s}\,|_{\mathbf{x}=\mathbf{x}_1} &= \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} & 1 \end{bmatrix}^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} \\ 1 \end{bmatrix} \\
&+ \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} & 1 \end{bmatrix}^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} \\ 1 \end{bmatrix} \\
&= A + B \ \Rightarrow
\end{aligned}$$

$$
\begin{aligned}
A &= \lambda\left(n_x^2\lambda\frac{\partial \mathbf{n}}{\partial s}\bullet\mathbf{n}+n_y^2\lambda\frac{\partial \mathbf{n}}{\partial s}\bullet\mathbf{n}+n_z^2\lambda\frac{\partial \mathbf{n}}{\partial s}\bullet\mathbf{n}-n_x^2\mathbf{n}\bullet\frac{\partial \mathbf{P}_1}{\partial s}\right.\\
&\quad \left.-n_y^2\mathbf{n}\bullet\frac{\partial \mathbf{P}_1}{\partial s}-n_z^2\mathbf{n}\bullet\frac{\partial \mathbf{P}_1}{\partial s}\right)=\\
&= \lambda\left(\lambda\frac{\partial \mathbf{n}}{\partial s}\bullet\mathbf{n}(n_x^2+n_y^2+n_z^2)-\mathbf{n}\bullet\frac{\partial \mathbf{P}_1}{\partial s}(n_x^2+n_y^2+n_z^2)\right)=\\
&= \lambda\left\|\mathbf{n}\right\|^2\mathbf{n}\left(\lambda\frac{\partial \mathbf{n}}{\partial s}-\frac{\partial \mathbf{P}_1}{\partial s}\right)
\end{aligned}
$$

$$
\begin{aligned}
B &= \lambda\left(\lambda\frac{\partial \mathbf{n}}{\partial s}\bullet\mathbf{n}\left\|\mathbf{n}\right\|^2-\mathbf{n}\bullet\frac{\partial \mathbf{P}_1}{\partial s}\left\|\mathbf{n}\right\|^2\right)\\
&= A\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s}\mid_{\mathbf{x}=\mathbf{x}_1}= 2\lambda\left\|\mathbf{n}\right\|^2\mathbf{n}\left(\lambda\frac{\partial \mathbf{n}}{\partial s}-\frac{\partial \mathbf{P}_1}{\partial s}\right).
\end{aligned}
$$

2. Derivatives of the distance d_{c_1} to the line aligned with $\mathbf{m}_1$-axis:

$$
\begin{aligned}
\frac{\partial d_{c_1}(\mathbf{x},\mathbf{S})}{\partial s}\mid_{\mathbf{x}=\mathbf{x}_1} &= 2\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{bmatrix}^T T_r^T L_c\frac{\partial T_r(\mathbf{S})}{\partial s}\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{bmatrix}=\\
&= 2\lambda\begin{bmatrix}0 & 0 & \left\|\mathbf{n}\right\|^2 & 0\end{bmatrix}\begin{bmatrix}\frac{\partial \mathbf{m}_1^T}{\partial s} & -\frac{\partial \mathbf{m}_1^T}{\partial s}\bullet\mathbf{P}_1-\mathbf{m}_1\bullet\frac{\partial \mathbf{P}_1^T}{\partial s}\\ \frac{\partial \mathbf{m}_2^T}{\partial s} & -\frac{\partial \mathbf{m}_2^T}{\partial s}\bullet\mathbf{P}_1-\mathbf{m}_2\bullet\frac{\partial \mathbf{P}_1^T}{\partial s}\\ \frac{\partial \mathbf{n}^T}{\partial s} & -\frac{\partial \mathbf{n}^T}{\partial s}\bullet\mathbf{P}_1-\mathbf{n}\bullet\frac{\partial \mathbf{P}_1^T}{\partial s}\\ \mathbf{0}^T & 0\end{bmatrix}\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{bmatrix}\\
&= 2\lambda\left\|\mathbf{n}\right\|^2\begin{bmatrix}\frac{\partial \mathbf{n}^T}{\partial s} & -\frac{\partial \mathbf{n}^T}{\partial s}\bullet\mathbf{P}_1-\mathbf{n}\bullet\frac{\partial \mathbf{P}_1^T}{\partial s}\end{bmatrix}\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{bmatrix}\\
&= 2\lambda\left\|\mathbf{n}\right\|^2\mathbf{n}\left(\lambda\frac{\partial \mathbf{n}}{\partial s}-\frac{\partial \mathbf{P}_1}{\partial s}\right).
\end{aligned}
$$

3. Derivatives of the distance d_s to the point $\mathbf{P}_1$:

$$
\begin{aligned}
\frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial s}\mid_{\mathbf{x}=\mathbf{x}_1} &= 2\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{bmatrix}^T T_r^T L_s\frac{\partial T_r(\mathbf{S})}{\partial s}\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{bmatrix}=\\
&= 2\lambda\left\|\mathbf{n}\right\|^2\begin{bmatrix}\frac{\partial \mathbf{n}^T}{\partial s} & -\frac{\partial \mathbf{n}^T}{\partial s}\bullet\mathbf{P}_1-\mathbf{n}\bullet\frac{\partial \mathbf{P}_1^T}{\partial s}\end{bmatrix}\begin{bmatrix}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{bmatrix}\\
&= 2\lambda\left\|\mathbf{n}\right\|^2\mathbf{n}\left(\lambda\frac{\partial \mathbf{n}}{\partial s}-\frac{\partial \mathbf{P}_1}{\partial s}\right).
\end{aligned}
$$

4. Derivatives of the distance $d_{c_2}$to the line aligned with $\mathbf{m}_2$-axis:
   In this case cylinder is aligned with $\mathbf{m}_2$-axis in the local coordinate frame, and transformation matrix becomes:

$$T_r = \begin{bmatrix} \mathbf{m}_2^T & -\mathbf{m}_2 \bullet \mathbf{P}_1 \\ \mathbf{m}_1^T & -\mathbf{m}_1 \bullet \mathbf{P}_1 \\ -\mathbf{n}^T & \mathbf{n} \bullet \mathbf{P}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}, \; T_r^T = \begin{bmatrix} \mathbf{m}_2 & \mathbf{m}_1 & -\mathbf{n} & \mathbf{0} \\ -\mathbf{m}_2 \bullet \mathbf{P}_1 & -\mathbf{m}_1\mathbf{P}_1 & \mathbf{n} \bullet \mathbf{P}_1 & 1 \end{bmatrix}$$

$$\begin{aligned} \frac{\partial d_{c_2}(\mathbf{x}, \mathbf{S})}{\partial s} |_{\mathbf{x}=\mathbf{x}_1} &= 2 \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} & 1 \end{bmatrix}^T T_r^T L_c \frac{\partial T_r(\mathbf{S})}{\partial s} \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} \\ 1 \end{bmatrix} = \\ &= 2\lambda \left\| \mathbf{n} \right\|^2 \begin{bmatrix} \frac{\partial \mathbf{n}^T}{\partial s} & -\frac{\partial \mathbf{n}^T}{\partial s} \bullet \mathbf{P}_1 - \mathbf{n} \bullet \frac{\partial \mathbf{P}_1^T}{\partial s} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} \\ 1 \end{bmatrix} \\ &= 2\lambda \left\| \mathbf{n} \right\|^2 \mathbf{n} \left(\lambda \frac{\partial \mathbf{n}}{\partial s} - \frac{\partial \mathbf{P}_1}{\partial s} \right). \end{aligned}$$

First order derivatives over the independent variables x, y and z in the points $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n}$:

1. Derivatives of the distance to plane d_p function:

$$\begin{aligned} \frac{\partial d_p(\mathbf{x}, \mathbf{S})}{\partial x_i} |_{\mathbf{x}=\mathbf{x}_1} &= 2 \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} & 1 \end{bmatrix}^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda \left\| \mathbf{n} \right\|^2 \begin{bmatrix} \mathbf{n}^T & - \mathbf{n} \bullet \mathbf{P}_1 \end{bmatrix}^T \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda n_i \left\| \mathbf{n} \right\|^2, n_i \in \{n_x, n_y, n_z\}, x_i \in \{x, y, z\} \end{aligned}$$

2. Derivatives of the distance $d_{c_1}$to the line aligned with $\mathbf{m}_1$-axis:

$$\begin{aligned} \frac{\partial d_{c_1}(\mathbf{x}, \mathbf{S})}{\partial x_i} |_{\mathbf{x}=\mathbf{x}_1} &= 2 \begin{bmatrix} \mathbf{P}_1 + \lambda\mathbf{n} & 1 \end{bmatrix}^T T_r^T L_c T_r \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda \left\| \mathbf{n} \right\|^2 \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{m}_1^T & -\mathbf{m}_1 \bullet \mathbf{P}_1 \\ \mathbf{m}_2^T & -\mathbf{m}_2 \bullet \mathbf{P}_1 \\ \mathbf{n}^T & -\mathbf{n} \bullet \mathbf{P}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda \left\| \mathbf{n} \right\|^2 \begin{bmatrix} \mathbf{n}^T & - \mathbf{n} \bullet \mathbf{P}_1 \end{bmatrix}^T \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda n_i \left\| \mathbf{n} \right\|^2, n_i \in \{n_x, n_y, n_z\}, x_i \in \{x, y, z\} \end{aligned}$$

3. Derivatives of the distance d_s to the point $\mathbf{P}_1$:

$$\begin{aligned}\frac{\partial d_s(\mathbf{x},\mathbf{S})}{\partial x_i}\mid_{\mathbf{x}=\mathbf{x}_1} &= 2\left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{array}\right]^T T_r^T L_s T_r \frac{\partial \mathbf{x}}{\partial x_i} = \\ &= 2\lambda n_i \left\|\mathbf{n}\right\|^2, n_i \in \{n_x, n_y, n_z\}, x_i \in \{x, y, z\}\end{aligned}$$

4. Derivatives of the distance $d_{c_2}$to the line aligned with $\mathbf{m}_2$-axis:

$$\begin{aligned}\frac{\partial d_{c_2}(\mathbf{x},\mathbf{S})}{\partial s}\mid_{\mathbf{x}=\mathbf{x}_1} &= 2\left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{array}\right]^T T_r^T L_c T_r \left[\begin{array}{c}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{array}\right] = \\ &= 2\left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{array}\right]^T \left[\begin{array}{cccc}\mathbf{m}_2 & \mathbf{m}_1 & -\mathbf{n} & \mathbf{0}\\ -\mathbf{m}_2\bullet\mathbf{P}_1 & -\mathbf{m}_1\mathbf{P}_1 & \mathbf{n}\bullet\mathbf{P}_1 & 1\end{array}\right] L_c T_r \left[\begin{array}{c}\mathbf{P}_1+\lambda\mathbf{n}\\ 1\end{array}\right] = \\ &= 2\lambda\left\|\mathbf{n}\right\|^2\left[\begin{array}{cccc}0 & 0 & -1 & 0\end{array}\right]\left[\begin{array}{cc}\mathbf{m}_2^T & -\mathbf{m}_2\bullet\mathbf{P}_1\\ \mathbf{m}_1^T & -\mathbf{m}_1\bullet\mathbf{P}_1\\ -\mathbf{n}^T & \mathbf{n}\bullet\mathbf{P}_1\\ \mathbf{0}^T & 1\end{array}\right]\frac{\partial\mathbf{x}}{\partial x_i} = \\ &= 2\lambda\left\|\mathbf{n}\right\|^2\left[\begin{array}{cc}\mathbf{n}^T & -\mathbf{n}\bullet\mathbf{P}_1\end{array}\right]^T\frac{\partial\mathbf{x}}{\partial x_i} = 2\lambda n_i\left\|\mathbf{n}\right\|^2, n_i\in\{n_x,n_y,n_z\}, x_i\in\{x,y,z\}\end{aligned}$$

CASE 3: Let us consider all the points $\mathbf{x}_1 = \mathbf{P}_1 + \lambda\mathbf{n} + \eta\mathbf{m}_1$ on the connection of regions $reg1$, $reg4$ where those regions actually meet. This is actually connection between cylinder and the plane along the edge $\mathbf{m}_1$ aligned with the x-axis. We first compute first order derivatives over parameters at these points for each of the two distance functions, and then first order derivatives over the independent variables x, y and z.

First order derivatives over the parameters in the points $\mathbf{x}_2 = \mathbf{P}_1 + \lambda\mathbf{n} + \eta\mathbf{m}_1$:

1. Derivatives of the distance to plane d_p function:

$$\begin{aligned}\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s}\mid_{\mathbf{x}=\mathbf{x}_2} &= \left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 & 1\end{array}\right]^T\frac{\partial L_1(\mathbf{S})}{\partial s}L_2\left[\begin{array}{c}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1\\ 1\end{array}\right] + \\ &= \left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 & 1\end{array}\right]^T L_1\frac{\partial L_2(\mathbf{S})}{\partial s}\left[\begin{array}{c}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1\\ 1\end{array}\right] = \\ &= A' + B' \Rightarrow\end{aligned}$$

$$
\begin{aligned}
A' &= \left[\begin{array}{cc} \mathbf{P}_1+\lambda\mathbf{n} & 1 \end{array}\right]^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n} \\ 1 \end{array}\right] + \\
&\left[\begin{array}{cc} \mathbf{P}_1+\lambda\mathbf{n} & 1 \end{array}\right]^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T + \\
&\left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n} \\ 0 \end{array}\right] + \\
&\left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T \frac{\partial L_1(\mathbf{S})}{\partial s} L_2 \left[\begin{array}{cc} \eta\mathbf{m}_1 & 1 \end{array}\right]^T = \\
&= \lambda \left\|\mathbf{n}\right\|^2 \mathbf{n} \left(\lambda \frac{\partial \mathbf{n}}{\partial s} - \frac{\partial \mathbf{P}_1}{\partial s}\right) + 0 + \lambda\eta\mathbf{m}_1 \frac{\partial \mathbf{n}}{\partial s} \left\|\mathbf{n}\right\|^2 + 0 = \\
&= \lambda \left\|\mathbf{n}\right\|^2 \left(\frac{\partial \mathbf{n}}{\partial s}\left(\lambda\mathbf{n}+\eta\mathbf{m}_1\right) - \mathbf{n}\frac{\partial \mathbf{P}_1}{\partial s}\right) \\
B' &= \left[\begin{array}{cc} \mathbf{P}_1+\lambda\mathbf{n} & 1 \end{array}\right]^T L1 \frac{\partial L_2(\mathbf{S})}{\partial s} \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n} \\ 1 \end{array}\right] + \\
&\left[\begin{array}{cc} \mathbf{P}_1+\lambda\mathbf{n} & 1 \end{array}\right]^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} \left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T = \\
&= \left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n} \\ 0 \end{array}\right] + \\
&\left[\begin{array}{cc} \eta\mathbf{m}_1 & 0 \end{array}\right]^T L_1 \frac{\partial L_2(\mathbf{S})}{\partial s} L_2 \left[\begin{array}{cc} \eta\mathbf{m}_1 & 1 \end{array}\right]^T = \\
&= \lambda \left\|\mathbf{n}\right\|^2 \mathbf{n} \left(\lambda \frac{\partial \mathbf{n}}{\partial s} - \frac{\partial \mathbf{P}_1}{\partial s}\right) + \\
&+ \lambda\eta\mathbf{m}_1 \frac{\partial \mathbf{n}}{\partial s} \left\|\mathbf{n}\right\|^2 + 0 + 0 = A'
\end{aligned}
$$

$$
\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial s} \mid_{\mathbf{x}=\mathbf{x}_1} = 2\lambda \left\|\mathbf{n}\right\|^2 \left(\frac{\partial \mathbf{n}}{\partial s}\left(\lambda\mathbf{n}+\eta\mathbf{m}_1\right) - \mathbf{n}\frac{\partial \mathbf{P}_1}{\partial s}\right)
$$

2. Derivatives of the distance d_{c_1} to the line aligned with $\mathbf{m}_1$-axis:

$$
\begin{aligned}
\frac{\partial d_{c_1}(\mathbf{x},\mathbf{S})}{\partial s} \mid_{\mathbf{x}=\mathbf{x}_2} &= 2\left[\begin{array}{cc} \mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 & 1 \end{array}\right]^T T_r^T L_c \frac{\partial T_r(\mathbf{S})}{\partial s} \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 \\ 1 \end{array}\right] = \\
&= 2\lambda \left[\begin{array}{cccc} 0 & 0 & \left\|\mathbf{n}\right\|^2 & 0 \end{array}\right] L_c \frac{\partial T_r(\mathbf{S})}{\partial s} \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 \\ 1 \end{array}\right] = \\
&= 2\lambda \left\|\mathbf{n}\right\|^2 \left[\begin{array}{cc} \frac{\partial \mathbf{n}^T}{\partial s} & -\frac{\partial \mathbf{n}^T}{\partial s} \bullet \mathbf{P}_1 - \mathbf{n} \bullet \frac{\partial \mathbf{P}_1^T}{\partial s} \end{array}\right] \left[\begin{array}{c} \mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 \\ 1 \end{array}\right] = \\
&= 2\lambda \left\|\mathbf{n}\right\|^2 \left(\frac{\partial \mathbf{n}}{\partial s}\left(\lambda\mathbf{n}+\eta\mathbf{m}_1\right) - \mathbf{n}\frac{\partial \mathbf{P}_1}{\partial s}\right).
\end{aligned}
$$

First order derivatives over the independent variables x, y and z in the points $\mathbf{x}_2 = \mathbf{P}_1 + \lambda\mathbf{n} + \eta\mathbf{m}_1$:

1. Derivatives of the distance to plane d_p function:

$$\begin{aligned}
\frac{\partial d_p(\mathbf{x},\mathbf{S})}{\partial x_i}\,|_{\mathbf{x}=\mathbf{x}_2} &= 2\left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 & 1\end{array}\right]^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} = \\
&= \left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{array}\right]^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} + \left[\begin{array}{cc}\eta\mathbf{m}_1 & 0\end{array}\right]^T L_1 L_2 \frac{\partial \mathbf{x}}{\partial x_i} = \\
&= 2\lambda\left\|\mathbf{n}\right\|^2 \left[\begin{array}{cc}\mathbf{n}^T & -\,\mathbf{n}\bullet\mathbf{P}_1\end{array}\right]^T \frac{\partial \mathbf{x}}{\partial x_i} + 0 = \\
&= 2\lambda n_i \left\|\mathbf{n}\right\|^2, n_i \in \{n_x, n_y, n_z\}, x_i \in \{x, y, z\}
\end{aligned}$$

2. Derivatives of the distance d_{c_1} to the line aligned with $\mathbf{m}_1$-axis:

$$\begin{aligned}
\frac{\partial d_{c_1}(\mathbf{x},\mathbf{S})}{\partial x_i}\,|_{\mathbf{x}=\mathbf{x}_2} &= 2\left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n}+\eta\mathbf{m}_1 & 1\end{array}\right]^T T_r^T L_c T_r \frac{\partial \mathbf{x}}{\partial x_i} = \\
&= \left[\begin{array}{cc}\mathbf{P}_1+\lambda\mathbf{n} & 1\end{array}\right]^T T_r^T L_c T_r \frac{\partial \mathbf{x}}{\partial x_i} + \left[\begin{array}{cc}\eta\mathbf{m}_1 & 0\end{array}\right]^T T_r^T L_c T_r \frac{\partial \mathbf{x}}{\partial x_i} = \\
&= 2\lambda\left\|\mathbf{n}\right\|^2 \left[\begin{array}{cc}\mathbf{n}^T & -\,\mathbf{n}\bullet\mathbf{P}_1\end{array}\right]^T \frac{\partial \mathbf{x}}{\partial x_i} \\
&+ \left[\begin{array}{cccc}\left\|\mathbf{m}_1\right\|^2 & 0 & 0 & 0\end{array}\right] \left[\begin{array}{cccc}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{array}\right] T_r \frac{\partial \mathbf{x}}{\partial x_i} = \\
&= 2\lambda n_i \left\|\mathbf{n}\right\|^2, n_i \in \{n_x, n_y, n_z\}, x_i \in \{x, y, z\}
\end{aligned}$$

Bibliography

[1] BAJAJ, C. Surface fitting with implicit algebraic surface patches. In *H. Hagen (ed.) Topics in Surface Modeling*. SIAM Publications, Philadelphia, 1992, pp. 23–52.

[2] BAJAJ, C., AND IHM, I. C^1 Smoothing of Polyhedra with Implicit Algebraic Splines. *Computer Graphics, SIGGRAPH Proceedings 26*, 2 (1992), 79–88.

[3] BAJAJ, C. L., BERNARDINI, F., AND XU, G. Automatic reconstruction of surfaces and scalar fields from 3D scans. *Computer Graphics 29*, Annual Conference Series (1995), 109–118.

[4] BARDINET, E., COHEN, L., AND AYACHE, N. A Parametric Deformable Model to Fit Unstructured 3D Data. *Computer Vision and Image Understanding 71*, 1 (1998), 39–54.

[5] BARR, A. H. Superquadrics and angle-preserving transformations. In *IEEE Computer Graphics and Applications* (1981), vol. 1, pp. 11–23.

[6] BEATSON, R., AND GREENGARD, L. A short course on fast multipole methods. In *Wavelets, Multilevel Methods and Elliptic PDEs*, M. Ainsworth, J. Levesley, W. Light, and M. Marletta, Eds. Oxford University Press, 1997, pp. 1–37.

[7] BLANZ, V., AND VETTER, T. A Morphable Model for The Synthesis of 3–D Faces. In *Computer Graphics, SIGGRAPH Proceedings* (Los Angeles, CA, August 1999).

[8] BLINN, J. F. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics 1*, 3 (1982), 235–256.

[9] BLOOMENTHAL, J. Calculation of reference frames along a space curve. In *Graphics Gems*, A. Glassner, Ed. Academic Press, Cambridge, MA, 1990, pp. 567–571.

[10] BLOOMENTHAL, J. Bulge elimination for convolution surfaces. In *Computer Graphics Forum* (1997).

[11] BLOOMENTHAL, J., AND SHOEMAKE, K. Convolution Surfaces. *Computer Graphics, SIGGRAPH Proceedings 25*, 4 (1991), 251–256.

[12] BLOOMENTHAL, J., AND WYVILL, B. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997.

[13] BOOKSTEIN, F. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 2 (1989), 567–585.

[14] BOOKSTEIN, F. Thin-plate splines and the atlas problem for biomedical images. In *Information Processing in Images* (1991), pp. 326–342.

[15] BOURKE, P. Implicit surfaces: http://astronomy.swin.edu.au/ pbourke/modelling/implicitsurf. http://astronomy.swin.edu.au/ pbourke/modelling/implicitsurf/, June 1997.

[16] CATMULL, E., AND CLARK, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Added Design 10* (1978), 350–355.

[17] CHANG, Y., AND ROCKWOOD, A. P. A Generalized de Casteljau Approach to 3D Free-form Deformation. *Computer Graphics, SIGGRAPH Proceedings* (1994), 257–260.

[18] CHENG, K.-S. D., WANG, W., QIN, H., WONG, K.-Y. K., YANG, H., AND LIU, Y. Fitting subdivision surfaces to unorganized point data using sdm. In *Pacific Conference on Computer Graphics and Applications* (2004), pp. 16–24.

[19] CIPOLLA, R., AND BLAKE, A. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision 9*, 2 (1992), 83–112.

[20] COHEN, I., COHEN, L. D., AND AYACHE, N. Introducing new deformable surfaces to segment 3D images. In *Conference on Computer Vision and Pattern Recognition* (1991), pp. 738–739.

[21] COHEN, L., , AND COHEN, I. Deformable models for 3-d medical images using finite elements and balloons. In *Conference on Computer Vision and Pattern Recognition* (1992), pp. 592–598.

[22] COQUILLART, S. Extended Free-Form Deformation: A sculpturing Tool for 3D Geometric Modeling. *Computer Graphics, SIGGRAPH Proceedings 24*, 4 (1990), 187–196.

[23] D. METAXAS, D. T. Constrained deformable superquadrics and nonrigid motion tracking. In *Conference on Computer Vision and Pattern Recognition* (Lahaina, HI, 1991), pp. 337–343.

[24] D. METAXAS, D. T. Constrained deformable superquadrics and nonrigid motion tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 6 (1993), 580–591.

[25] DE BOOR, C. Quasiinterpolants and approximation power of multivariate splines. In *Computations of curves and surfaces* (Dordrecht, Netherlands, 1990), Kluwer, pp. 313–345.

[26] DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proceedings of SIGGRAPH 96* (August 1996), 11–20. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[27] DECARLO, D., AND METAXAS, D. The Integration of Optical Flow and Deformable Models with Applications to Human Face Shape and Motion Estimation. In *Conference on Computer Vision and Pattern Recognition* (1996), pp. 231–238.

[28] DECARLO, D., AND METAXAS, D. Shape evolution with structural and topological changes using blending. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 11 (1998), 1186–1205.

[29] DELINGETTE, H., HEBERT, M., AND IKEUCHI, K. Deformable surfaces: A free-form shape representation. In *Proc. SPIE Geometric Methods in Computer Vision* (1991), vol. 1570, pp. 21–30.

[30] DELINGETTE, H., HEBERT, M., AND IKEUCHI, K. Shape representation and image segmentation using deformable surfaces. *Image Vision Computing 10*, 3 (1992), 132–144.

[31] DESBRUN, M., AND GASCUEL, M. Animating Soft Substances with Implicit Surfaces. *Computer Graphics, SIGGRAPH Proceedings* (1995), 287–290.

[32] DEWAELE, G., DEVERNAY, F., AND HORAUD, R. Hand Motion from 3D Point Trajectories and a Smooth Surface Model. In *European Conference on Computer Vision* (Prague, Czech Republic, May 2004).

[33] DIERCKX, P. *Curve and surface fitting with splines*. Oxford University Press, Inc., 1993.

[34] DIMITRIJEVIĆ, M., ILIĆ, S., AND FUA, P. Accurate Face Models from Uncalibrated and Ill-Lit Video Sequences. In *Conference on Computer Vision and Pattern Recognition* (Washington, DC, June 2004).

[35] DION, JR., D., LAURENDEAU, D., AND BERGEVIN, R. Generalized cylinder extraction in range images. In *3DIM97* (1997), pp. 6 – Geometric Processing.

[36] DOO, D., AND SABIN, M. Behaviour of Recursive Division Surfaces Near Extraordinary Points. *Computer Added Design 10*, 6 (1978), 356–360.

[37] DRUMMOND, T., AND CIPOLLA, R. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (july 2002), 932–946.

[38] ECK, M., AND HOPPE, H. Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *Computer Graphics, SIGGRAPH Proceedings* (1996), pp. 325–334.

[39] ET AL., H. N. Object Modeling by Distribution Function and a Method of Image Generation. In *Journal od papers given at the Electronic Communication Conf.* (1985).

[40] ET AL., J. C. C. Reconstruction and representation of 3d objects with radial basis functions. In *Computer Graphics, SIGGRAPH Proceedings* (2001), vol. 2.

[41] ET AL., J. C. C. Smooth surface reconstruction from noisy range data. In *ACM GRAPHITE* (Melbourne, Australia, 2003), pp. 119–126.

[42] FARIN, G. Surface over Dirichlet Tessellations. *Computer Aided Design 7* (1990), 281–292.

[43] FAUGERAS, O. A Few Steps Toward Artificial 3D Vision. Rapport de Recherche 790, INRIA, Feb. 1988.

[44] FUA, P. Combining Stereo and Monocular Information to Compute Dense Depth Maps that Preserve Depth Discontinuities. In *International Joint Conference on Artificial Intelligence* (Sydney, Australia, August 1991), pp. 1292–1298.

[45] FUA, P. A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. *Machine Vision and Applications 6*, 1 (Winter 1993), 35–49.

[46] FUA, P. From Multiple Stereo Views to Multiple 3–D Surfaces. *International Journal of Computer Vision 24*, 1 (August 1997), 19–35.

[47] FUA, P. Regularized Bundle-Adjustment to Model Heads from Image Sequences without Calibration Data. *International Journal of Computer Vision 38*, 2 (July 2000), 153–171.

[48] FUA, P., AND LECLERC, Y. G. Combining Stereo, Shading and Geometric Constraints for Surface Reconstruction from Multiple Views. In *SPIE Workshop on Geometric Methods in Computer Vision* (San Diego, CA, July 1993), pp. 113–123.

[49] FUA, P., AND MICCIO, C. Animated Heads from Ordinary Images: A Least Squares Approach. *Computer Vision and Image Understanding 75*, 3 (September 1999), 247–259.

[50] GAVRILA, D., AND DAVIS, L. 3-D Model-based Tracking of Human Upper Body Movement: a Multi-View Approach. In *IEEE Interntational Symposium on Computer Vision* (November 1995), IEEE Computer Society Press, pp. 253–258.

[51] GUPTA, A., AND BAJCSY, R. Volumetric segmentation of range images of 3d objects using superquadrics models. *Computer Vision, Graphics, and Image Processing 3*, 58 (1993), 302–326.

[52] H. QIN, D. T. D-nurbs: A physics-based framework for geometric designe. In *IEEE Transactions on Visualization and Computer Graphics* (March 1996), vol. 2, pp. 85–96.

[53] HAN, S., AND MEDIONI, G. Reconstructing free-form surfaces from sparse data. In *International Conference on Pattern Recognition* (1996), p. A70.2.

[54] HARTLEY, R., AND ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[55] HILL, F. S. *Computer Graphics using OpenGL*. Prentice Hall, 2000.

[56] HILTON, A., BERESFORD, D., GENTILS, T., SMITH, R., AND SUN, W. Virtual People: Capturing Human Models to Populate Virtual Worlds. In *Computer Animation* (Geneva, Switzerland, May 1999).

[57] HOPPE, H., DEROSE, T., DUCHAMP, T., HALSTEAD, M., JUN, H., MCDONALD, J., SCHWEITZER, J., AND STUETZLE, W. Piecewise Smooth Surface Reconstruction. In *Computer Graphics, SIGGRAPH Proceedings* (1994), pp. 295–302.

[58] HORAUD, R., AND BRADY, M. On the geometric interpretation of image contours. *Artif. Intell. 37*, 1-3 (1988), 333–353.

[59] ILIĆ, S., AND FUA, P. Using Dirichlet Free Form Deformation to Fit Deformable Models to Noisy 3-D Data. In *European Conference on Computer Vision* (Copenhagen, Denmark, May 2002).

[60] ILIĆ, S., AND FUA, P. From Explicit to Implicit Surfaces for Visualization, Animation and Modeling. In *ISPRS workshop on Visualization and Animation of Reality-based 3D Models* (Vulpera, Switzerland, February 2003).

[61] ILIĆ, S., AND FUA, P. Generic Deformable Implicit Mesh Models for Automated Reconstruction. In *ICCV Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis* (Nice, France, October 2003).

[62] ILIĆ, S., AND FUA, P. Implicit Mesh Models for Modeling and Tracking. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, June 2003).

[63] ILIĆ, S., SALZMANN, M., AND FUA, P. Implicit Mesh Make for Better Silhouettes. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005).

[64] J. C. CARR, W. R. F., AND BEATSON, R. K. Surface Interpolation with Radial Basis Functions for Medical Imaging. *IEEE Transactions on Medical Imaging 16*, 1 (1997), 96–107.

[65] JAKLIČ, A., LEONARDIS, A., AND SOLINA, F. *Segmentation and Recovery of Superquadrics*, vol. 20 of *Computational imaging and vision*. Kluwer, Dordrecth, 2000. ISBN 0-7923-6601-8.

[66] JEONG, W. K., AND KIM, C. H. Direct reconstruction of displaced subdivision surface from unorganized points. In *Pacific Conference on Computer Graphics and Applications* (October 2001), pp. 160–169.

[67] JOJIĆ., N., AND HUANG, T. S. Computer vision and graphics techniques for modeling dressed humans. In *Conference of Computer Vision and Computer Graphics*, S. F. Leonardis A. and B. R., Eds. Kluwer, 2000, pp. 179–200.

[68] KALRA, P., MANGILI, A., THALMANN, N. M., AND THALMANN, D. Simulation of Facial Muscle Actions Based on Rational Free Form Deformations. In *Eurographics* (1992).

[69] KANG, S., AND JONES, M. Appearance-based structure from motion using linear classes of 3-d models. *International Journal of Computer Vision 49*, 1 (2002), 5–22.

[70] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active Contour Models. *International Journal of Computer Vision 1*, 4 (1988), 321–331.

[71] KOLLER, T., GERIG, G., SZEKELY, G., AND DETTWILER, D. Multiscale detection of curvilinear structures in 2d and 3d image data. In *International Conference on Computer Vision* (1995), pp. 864–869.

[72] KUTULAKOS, K., AND SEITZ, S. A Theory of Shape by Space Carving. *International Journal of Computer Vision 38*, 3 (July 2000), 197–216.

[73] LEE, A., MORETON, H., AND HOPPE, H. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 85–94.

[74] LEE, W., GU, J., AND THALMANN, N. M. Generating Animatable 3D Virtual Humans from Photographs. In *Computer Graphics Forum, Eurographics* (Interlaken, Switzerland, August 2000), pp. 1–10.

[75] LEE, W., AND THALMANN, N. M. Fast Head Modelling for Animation. *Journal Image and Vision Computing 18*, 4 (August 2000).

[76] LEONARDIS, A., JAKLIČ, A., AND SOLINA, F. Superquadrics for segmentation and modeling range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 11 (1997), 1289–1295.

[77] LITKE, N., LEVIN, A., AND SCHRÖDER, P. Fitting subdivision surfaces. In *Proceedings of the conference on Visualization '01* (2001), IEEE Computer Society, pp. 319–324.

[78] LIU, J., MUNDY, J., FORSYTH, D., AND ZISSERMAN, A. Efficient recognition of rotationally symmetric surface and straight homogeneous generalized cylinders. *Conference on Computer Vision and Pattern Recognition* (1993), 123–128.

[79] LOOP, C. Smooth Subdivision Surfaces Based on Triangles. Master thesis, Department of Mathematics, University of Utah, 1987.

[80] LOWE, D. G. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 441-450 (1991).

[81] MA, W., AND KRUTH, J. P. Parametrization of randomly measured points for least square fitting of b-spline curve and surfaces. *Computer Added Design 1*, 27 (1995), 663–675.

[82] METAXAS, D., AND TERZOPOULOS, D. Recursive estimation of shape and nonrigid motion. In *In IEEE Workshop on Visual Motion* (October 1991), pp. 306–311.

[83] METAXAS, D., AND TERZOPOULOS, D. Dynamic Deformations of Solid Primitives and Constraints. *Computer Graphics, SIGGRAPH Proceedings 26*, 2 (1992), 309–312.

[84] MOCCOZET, L., AND MAGNENAT-THALMANN, N. Dirichlet Free-Form Deformation and their Application to Hand Simulation. In *Computer Animation* (1997).

[85] NASTAR, C., AND AYACHE, N. Frequency-based nonrigid motion analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 11 (November 1996).

[86] PENTLAND, A. Automatic extraction of deformable part models. *International Journal of Computer Vision 4*, 2 (March 1990), 107–126.

[87] PENTLAND, A., HOROWITZ, B., AND SCLAROFF, S. Non-rigid motion and structure from contour. In *MOTION91* (1991), pp. 288–293.

[88] PENTLAND, A., AND WILLIAMS, J. Good Vibrations: Modal Dynamics for Graphics and Animation. *Computer Graphics, SIGGRAPH Proceedings 23*, 4 (1989), 215–222.

[89] PENTLAND, A. P. Perceptual organization and representation of natural form. *Artificial Inteligence 28*, 2 (1986), 293–331.

[90] PLÄNKERS, R., AND FUA, P. Articulated Soft Objects for Video-based Body Modeling. In *International Conference on Computer Vision* (Vancouver, Canada, July 2001), pp. 394–401.

[91] PLÄNKERS, R., AND FUA, P. Articulated Soft Objects for Multi-View Shape and Motion Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2003).

[92] PONCE, J., AND CHELBERG, D. Invariant properties of straight homogeneous generalised cylinders and their contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 9 (1989), 951–965.

[93] PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. *Numerical Recipes, the Art of Scientific Computing*. Cambridge U. Press, Cambridge, MA, 1986.

[94] RAJA, N. S., AND JAIN, A. K. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing 10*, 3 (1992), 179–190.

[95] ROSTEN, E., AND DRUMMOND, T. Rapid rendering of apparent contours of implicit surfaces for realtime tracking. In *British Machine Vision Conference* (Norwich, UK, 2003), vol. 2, pp. 719–728.

[96] ROY, S., AND COX, I. A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem. In *International Conference on Computer Vision* (Bombay, India, 1998), pp. 492–499.

[97] SATO, H., AND BINFORD, T. O. Finding and recovering shgc objects in an edge image. *Computer Vision, Graphics, and Image Processing 57*, 3 (1993), 346–358.

[98] SCHEIB, V., HABER, J., LIN, M. C., AND SEIDEL, H.-P. Efficient fitting and rendering of large scattered data sets using subdivision surfaces. *Computer Graphics Forum 21*, 3 (2002).

[99] SCLAROFF, S., AND PENTLAND, A. P. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 6 (1995), 545–561.

[100] SEDERBERG, T., AND PARRY, S. Free-Form Deformation of Solid Geometric Models. *Computer Graphics, SIGGRAPH Proceedings 20*, 4 (1986).

[101] SHAHROKNI, A., DRUMMOND, T., AND FUA, P. Texture Boundary Detection for Real-Time Tracking. In *European Conference on Computer Vision* (Prague, Czech Republic, May 2004), pp. Vol II: 566–577.

[102] SHAN, Y., LIU, Z., AND ZHANG, Z. Model-Based Bundle Adjustment with Application to Face Modeling. In *International Conference on Computer Vision* (Vancouver, Canada, July 2001).

[103] SHEN, X., AND SPANN, M. 3d shape modelling through a constrained estimation of a bicubic b-spline surface. In *British Machine Vision Conference* (1998), pp. xx–yy.

[104] SIBSON, R. A Vector Identity for the Dirichlet Tessellation. In *Math. Proc. Cambridge Philos. Soc.* (1980), pp. 151–155.

[105] SMINCHISESCU, C., AND TRIGGS, B. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research 22*, 6 (June 2003), 371–391.

[106] SOLINA, F., AND BAJCSY, R. Recovery of parametric models from range images: the case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 2 (1990), 131–147.

[107] STENGER, B., MENDONCA, P. R. S., AND CIPOLLA, R. Model-based 3D tracking of an Articulated Hand. In *Conference on Computer Vision and Pattern Recognition* (Kauai, USA, December 2001), pp. 310–315.

[108] STODDART, A., AND BAKER, M. Surface reconstruction and compression using multiresolution arbitrary topology g1 continuous splines. In *International Conference on Pattern Recognition* (1998), pp. Vol I: 788–791.

[109] SULLIVAN, S., AND PONCE, J. Automatic model construction and pose estimation from photographs using triangular splines. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 10 (October 1998), 1091–1097.

[110] SULLIVAN, S., SANDFORD, L., AND PONCE, J. Using geometric distance fits for 3–d. object modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 12 (December 1994), 1183–1196.

[111] SZELISKI, R., LAVAL, S., AND E. Matching anatomical surface with non-rigid deformations using octree-splines. *International Journal of Computer Vision 18*, 2 (1996), 171–186.

[112] SZELISKI, R., AND TERZOPOULOS, D. Physically based and probabilistic models for computer vision. *SPIE 1570* (1991), 140–152.

[113] TERZOPOULOS, D. On matching deformable models to images. In *Topical Meeting on Machine Vision, Technical Digest Series* (November 1986), vol. 12.

[114] TERZOPOULOS, D., AND METAXAS, D. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13* (1991), 703–714.

[115] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEICHER, K. Elastically Deformable Models. *Computer Graphics, SIGGRAPH Proceedings 21*, 4 (1987), 205–214.

[116] TERZOPOULOS, D., AND VASILESCU, M. Sampling and reconstruction with adaptive meshes. In *Conference on Computer Vision and Pattern Recognition* (1991), pp. 70–75.

[117] TERZOPOULOS, D., AND WITKIN, A. Physically Based Model with Rigid and Deformable Components. *IEEE Computer Graphics and Applications 8*, 6 (1988), 41–51.

[118] TURK, G., AND O'BRIEN, J. Shape transformation using variational implicit functions. In *Computer Graphics, SIGGRAPH Proceedings* (1999), vol. 33, pp. 335–342.

[119] TURK, G., AND O'BRIEN, J. Variational implicit surfaces. technical report git-gvu-99-15. Tech. rep., Graphics, Visualization, and Useability Center. Georgia Institute of Technology, 1999.

[120] TURK, H., AND O'BRIEN, J. Modelling with implicit surfaces that interpolate, 2000.

[121] VACCHETTI, L., LEPETIT, V., AND FUA, P. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *International Symposium on Mixed and Augmented Reality* (Arlington, VA, November 2004).

[122] WHAITE, P., AND FERRIE, F. From uncertanty to visual exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 10 (1991), 1038–1049.

[123] WILLIAMS, J., JOHNSTONE, J., AND WOLFF, L. Rational discrete generalized cylinders and their application to shape recovery in medical images. In *Conference on Computer Vision and Pattern Recognition* (1997), pp. 387–392.

[124] WONG, K.-Y. K., MENDONÇA, P. R. S., AND CIPOLLA, R. Reconstruction of surfaces of revolution from single uncalibrated views. *Image and Vision Computing 22*, 10 (2004), 829–836.

[125] WYVILL, G., AND WYVILL, B. Data Structure for Soft Objects. *The Visual Computer* (February 1986), 2(4)227–234.

[126] XU, C., AND PRINCE, J. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing 7*, 3 (Mar. 1998), 359–369.

[127] XU, Z. Subdivision surfaces: http://www.ke.ics.saitama-u.ac.jp/xuz/mid-subdivision.html.

[128] YUILLE, A. L., HALLINAN, P. W., AND COHEN, D. S. Feature extraction from faces using deformable templates. *International Journal of Computer Vision 8*, 2 (1992), 99–111.

[129] ZERROUG, M., AND NEVATIA, R. Three-dimensional descriptions based on the analysis of the invariant and quasi-invariant properties of some curved-axis generalized cylinders. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 3 (March 1996), 237–253.

[130] ZHAO, C., AND MOHR, R. Relative 3d regularized b-spline surface reconstruction through image sequences. In *European Conference on Computer Vision* (1994), pp. B:417–426.

Curriculum Vitae

General Data

Name Ilic Slobodan

Date of Birth 30. March 1973, Aleksinac, Serbia

Nationality Serbian

Languages Serbian (native), English and French fluent, Russian intermidiate

Education

2000 - 2005 PhD studies, Computer Vision Laboratory, EPFL, Switzerland

1999 - 2000 Doctoral School Program in Computer Science, EPFL

1992 - 1997 Electronic engineer with specialization in Computer Science, Faculty of Electronic Engineering, University of Nis, Yugoslavia

1988 - 1992 Mathematical Gymnasium "Bora Stankovic", Nis, Yugoslavia

Professional Experience

2000 - 2005 Research assistant, Computer Vision Lab, EPFL, Switzerland

1997 - 1999 Research assistant, Faculty of Electronic Engineering, University of Nis, Yugoslavia

1997 - 1998 VB and MS Access developer, Local TV station "TV5", Nis, Yugoslavia