Learning and Recovering 3D Surface Deformations

THÈSE N^{O} 4270 (2009)

PRÉSENTÉE LE 16 JANVIER 2009
À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE VISION PAR ORDINATEUR
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Mathieu SALZMANN

acceptée sur proposition du jury:

Prof. B. Moret, président du jury
Prof. P. Fua, directeur de thèse
Prof. P. Dillenbourg, rapporteur
Dr T. Papadopoulo, rapporteur
Dr L. Torresani, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2009

Abstract

Recovering the 3D deformations of a non-rigid surface from a single viewpoint has applications in many domains such as sports, entertainment, and medical imaging. Unfortunately, without any knowledge of the possible deformations that the object of interest can undergo, it is severely under-constrained, and extremely different shapes can have very similar appearances when reprojected onto an image plane.

In this thesis, we first exhibit the ambiguities of the reconstruction problem when relying on correspondences between a reference image for which we know the shape and an input image. We then propose several approaches to overcoming these ambiguities. The core idea is that some *a priori* knowledge about how a surface can deform must be introduced to solve them. We therefore present different ways to formulate that knownledge that range from very generic constraints to models specifically designed for a particular object or material.

First, we propose generally applicable constraints formulated as motion models. Such models simply link the deformations of the surface from one image to the next in a video sequence. The obvious advantage is that they can be used independently of the physical properties of the object of interest. However, to be effective, they require the presence of texture over the whole surface, and, additionally, do not prevent error accumulation from frame to frame.

To overcome these weaknesses, we propose to introduce statistical learning techniques that let us build a model from a large set of training examples, that is, in our case, known 3D deformations. The resulting model then essentially performs linear or non-linear interpolation between the training examples.

Following this approach, we first propose a linear global representation that models the behavior of the whole surface. As is the case with all statistical learning techniques, the applicability of this representation is limited by the fact that acquiring training data is far from trivial. A large surface can undergo many subtle deformations, and thus a large amount of training data must be available to build an accurate model. We therefore propose an automatic way of generating such training examples in the case of inextensible surfaces. Furthermore, we show that the resulting linear global models can be incorporated into a closed-form solution to the shape recovery problem. This lets us not only track deformations from frame to frame, but also reconstruct surfaces from individual images.

The major drawback of global representations is that they can only model the behavior of a specific surface, which forces us to re-train a new model for every new shape, even though it is made of a material observed before. To overcome this issue, and simultaneously reduce the amount of required training data, we propose local deformation models. Such models describe the behavior of small portions of a surface, and can be combined to form arbitrary global shapes. For this purpose, we study both linear and non-linear statistical learning methods, and show that, whereas the latter are better suited for traking deformations from frame to frame, the former can also be used for reconstruction from a single image.

Keywords: Computer Vision, Monocular 3D Reconstruction, Deformable Surfaces, Statistical Learning, Convex Optimization

Résumé

La reconstruction 3D d'une surface deformable en vision monoculaire est un problème pouvant s'appliquer à de nombreux domaines tels que le sport, le divertissement et l'imagerie médicale. Malheureusement, en l'absence totale de connaissance des deformations qu'un objet peut subir, ce problème est fortement sous-contraint, et des formes extrêmement différentes peuvent avoir des apparences très similaires une fois projetées dans une image.

Dans cette thèse, nous démontrons d'abord les ambiguïtés inhérentes à la reconstruction à partir de correspondences entre une image de référence pour laquelle nous connaissons la forme et une nouvelle image, puis nous proposons plusieurs façons de les surmonter. L'idée centrale est qu'une certaine forme de connaissance *a priori* de la manière dont une surface peut se deformer doit être introduite afin de résoudre ces ambiguïtés. Nous présentons donc plusieurs formulations de cette connaissance variant de modèles très généraux à d'autres spécifiquement construits pour un objet ou un matériau précis.

Dans un premier temps, nous proposons des contraintes générales formulées comme des modèles de mouvement. Ces modèles lient simplement les déformations d'une surface d'une image à la suivante dans une séquence vidéo. Un avantage évident de ces méthodes est qu'elles sont indépendentes de l'objet en particulier. Cependant, pour être efficaces, elles nécessitent la présence de texture sur toute la surface, et, de plus, présentent des risques d'accumuler les erreurs d'une image à l'autre.

Afin de résoudre ces faiblesses, nous proposons d'étudier des méthodes d'apprentissage statistique nous permettant de construire un modèle à partir d'une collection d'exemples d'apprentissage, c'est-à-dire, dans notre cas, de déformations 3D connues. Le modèle résultant consiste essentiellement en une interpolation linéaire ou non entre les données d'entraînement.

En suivant cette approche, nous proposons dans un premier temps une représentation globale linéaire qui modélise le comportement de la surface entière. Comme toutes les méthodes d'apprentissage, cette representation est limitée par le fait qu'acquérir des données d'entraînement peut s'avérer très compliqué. Une grande surface pouvant subir des déformations complexes, l'apprentissage du modèle requiert un grand nombre d'exemples. Nous proposons donc une méthode automatique pour générer de tels exemples dans le cas de surfaces inextensibles. De plus, nous montrons que les modèles globaux résultants peuvent être incorporés à une solution analytique au problème de reconstruction 3D. Ceci nous permet donc non seulement de suivre les déformations dans une séquence vidéo, mais aussi de retrouver la forme d'une surface à partir d'images individuelles, ce qui évite l'accumulation d'erreurs.

L'inconvénient majeur des représentations globales est qu'elles ne peuvent modéliser le comportement que d'une surface spécifique. Ainsi, un nouveau modèle doit être ré-entraîné pour chaque nouvelle surface, même si elle est faite d'un matériau observé précédemment. Afin de surmonter ce problème, et simultanément de réduire la quantité nécessaire de données d'entraînement, nous proposons des modèles de déformation locaux qui décrivent le comportement de petites portions d'une surface, et peuvent être assemblés de manière à former des surfaces globales de formes arbitraires. Pour ce faire, nous étudions des méthodes d'apprentissage statistique linéaires et non-linéaires, et montrons que, alors que les secondes sont mieux adaptées au suivi de déformations d'image en image, les premières peuvent aussi être utilisées pour la reconstruction à partir d'une image individuelle.

Mots-Clés: Vision par Ordinateur, Reconstruction 3D Monoculaire, Surfaces Deformables, Apprentissage Statistique, Optimisation Convexe

Acknowledgments

Many people have contributed more or less directly to this thesis. I am grateful to all of them for the many different ways they helped me in.

I would first like to thank Pascal Fua for giving me the opportunity to do this work and supporting me throughout the years. I am particularly grateful for his involvement in my research project. Having a professor who writes code for his Ph.D students is far from being common. I also greatly benefited from his excellent advice and attempted to learn from his amazing writing skills. I would be glad to keep on collaborating with him in the future.

Other researchers influenced my work on various topics. Among them, I would like to thank Richard Hartley who introduced me to the beauty of convex optimization, as well as to the beauty of Australia. Working with him allowed me to see my research from a new perspective. I would also like to thank Vincent Lepetit who, while Pascal was on sabbatical, gave a new impulse to my research, and with whom I kept, and hopefully will keep on collaborating. Last but not least, I am particularly grateful to Raquel Urtasun who contaminated me with her passion for research. She started as my advisor for my Master project and ended up as a great friend and a source of inspiration during my Ph.D. I look forward to working with her again.

I would like to thank the members of my jury for their useful comments and questions, and for showing interest and enthusiasm for my work.

I would also like to thank the current and former members of the CVLab for the friendly atmosphere and for their help. In particular, Lorna who was at the origin of the Master project that motivated me to stay at EPFL a little longer, Slobodan for his advice at the beginning of my Ph.D, Julien for some of the code I have been using for more than four years, Pascal for helping me so many times with the motion capture system, Francesc for his involvement in the closed-form reconstruction, and Aydin for making my work on deformable surfaces obsolete. Special thanks to Josiane who drove me around when my ankle was in a bad shape and without whom we would probably end up sleeping outside at every conference.

Many thanks to Trevor Darrell for inviting me at the MIT, and to the members of his group and other friends for making a great time of my stay there.

My gratitude also goes to the many friends that are not directly related to my work. I would first like to thank my friends Martin and Anouchka for their everlasting support and interest in my research. Many thanks to the other members of the now-dissolved band Skyyak: Michael, Christophe and Samuel for their friendship, the fun and the great music. Many thanks to Les Pires: Max, Jo, Erwann and Marc, and their Pirettes, to which I include

Juliette, for all the jeudimidis and Christmas fondues. Thanks to the members of FSG St-Prex for the fun and the sport that kept me in a good enough shape to climb up the stairs to my third floor office. Thanks to all my friends in Australia for the great three months I spent there.

I would also like to thank my family for their love and support, especially my parents for encouraging me in my unusual pre-EPFL career choice, without which I might have not ended up doing a Ph.D.

Finally and most importantly, I would like to thank Marie Julie for her love, support and patience, and for accepting my many absences. I am happy that you chose to follow me in this adventure, and I look forward to continuing the journey with you.

Contents

1 Introduction 25
1.1 Problem Definition . . . 25
1.2 Contributions . . . 27
1.2.1 Motion Models . . . 28
1.2.2 Global Models . . . 29
1.2.3 Local Models . . . 30
1.3 Thesis Outline . . . 31

2 Related Work 33
2.1 Physics-Based Methods . . . 33
2.1.1 The Finite Element Method . . . 34
2.1.2 Physics-based Methods for Computer Graphics . . . 34
2.1.3 The Computer Vision Approach . . . 35
2.1.4 Relations to our Work . . . 37
2.2 Learned Global Models . . . 37
2.2.1 Statistical Learning Methods . . . 37
2.2.2 Learned Models for Non-Rigid Modeling . . . 39
2.2.3 Relations to our Work . . . 40
2.3 Alternative Shape Constraints . . . 40
2.3.1 Using Control Points . . . 40
2.3.2 Imposing Additional Constraints . . . 41
2.4 Alternative Image Constraints . . . 41
2.4.1 Multiple Views . . . 42
2.4.2 Shape from Shading . . . 42
2.4.3 Shape from Silhouettes . . . 43
2.5 Summary . . . 44

3 The Monocular 3D Reconstruction Problem 45
3.1 3D-to-2D Correspondences . . . 45
3.2 Single-Image Ambiguities . . . 46
3.2.1 Ambiguities under Perspective Projection . . . 46
3.2.1.1 Projection of a 3D Surface Point . . . 47
3.2.1.2 Reconstructing a Single Facet . . . 47
3.2.1.3 Reconstructing the Whole Mesh . . . 49
3.2.1.4 Effective Rank . . . 50

3.2.2 Ambiguities under Piecewise Affine Projection 50
  3.2.2.1 Projection of a 3D Surface Point 50
  3.2.2.2 Reconstructing a Single Facet 51
  3.2.2.3 Reconstructing the Whole Mesh 52
3.3 Image Information in Practice 53
3.3.1 Texture 53
  3.3.1.1 Template Matching 54
  3.3.1.2 Obtaining Dense Correspondences 54
  3.3.1.3 2D Registration 55
3.3.2 Silhouettes 56
3.4 Conclusion 57

4 Motion Models 59
4.1 Linear Motion Model 59
  4.1.1 Constraining the Affine Reconstruction 59
  4.1.2 Constraining the Perspective Reconstruction 61
4.2 Experimental Results 63
  4.2.1 Synthetic Data 64
  4.2.2 Real Data 65
4.3 Convex Optimization 66
4.4 SOCP for Deformable Surfaces 67
  4.4.1 Correspondences as SOCP Constraints 67
  4.4.2 Additional Constraints 70
4.5 Experimental Results 71
  4.5.1 Synthetic Data 72
  4.5.2 Real Data 76
4.6 Conclusion 77

5 Global Deformation Models 81
5.1 Inextensible Triangulations 81
  5.1.1 Dofs of Inextensible Triangulations 82
  5.1.2 Angle-Based Parameterization 83
    5.1.2.1 Simple Triangulations 83
    5.1.2.2 Complex Triangulations 84
  5.1.3 Dimensionality Reduction 85
5.2 Shape Recovery by Tracking 87
5.3 Experimental Results 88
  5.3.1 Synthetic Data 89
  5.3.2 Real Images 90
    5.3.2.1 Tracking Results 90
    5.3.2.2 Tracking and Detection 93
5.4 Closed-Form Solution to Non-Rigid 3D Registration 98
  5.4.1 Linear Formulation 99

5.4.2 Inextensibility Constraints 99
5.4.3 Linear Deformation Model 103
5.5 Experimental Results 104
5.5.1 Synthetic Data 106
5.5.2 Real Images 107
5.6 Conclusion 109

6 Local Deformation Models **111**
6.1 Learning Local Models 111
6.1.1 Linear Local Models 112
6.1.2 The Gaussian Process Latent Variable Model 113
6.1.3 Probabilistic PCA as a GPLVM 114
6.1.4 Acquiring the Training Data 116
6.1.4.1 Real Data Acquisition 116
6.2 Global Models as Mixtures of Local Ones 117
6.2.1 PoE for Deformable Surfaces 118
6.2.2 Surface Boundary Effects 119
6.3 Monocular 3D Tracking 119
6.3.1 Image Likelihood 120
6.3.2 Optimization 120
6.4 Experimental Results 122
6.4.1 Local Models Validation 122
6.4.2 Synthetic Data 125
6.4.3 Real Sequences 127
6.5 Closed-Form 3D Reconstruction 132
6.5.1 Constraining the Reconstruction via Linear Local Models 132
6.5.2 Inextensibility Constraints 134
6.6 Experimental Results 135
6.6.1 Synthetic Data 135
6.6.2 Real Images 137
6.7 Conclusion 139

7 Comparative Results **141**
7.1 Competing Techniques 141
7.1.1 Smoothness Assumptions 141
7.1.2 Physics-based Deformation Modes 141
7.2 Comparing Deformation Spaces 144
7.2.1 Global Models 144
7.2.2 Local Models 147
7.3 Comparison of our Shape Recovery Techniques 148
7.3.1 Synthetic Data 151
7.3.2 Real Images 157

8 Conclusion **161**
8.1 Applications 162
8.2 Future Work 163

Curriculum Vitae **179**

List of Figures

1.1 Surface reconstruction can be applied to medical imaging. (a) Schematic representation of non-invasive surgery (b) Augmenting a scene with the reconstruction of a virtual liver [130]. 26

1.2 Many sports could benefit from 3D shape recovery. (a,b) Sailors are interested in knowing the shape of their own sails, or of the opponents' sails (c) Recovering the shape of skis during a race could help improving their design. 27

1.3 The entertainment industry could use 3D reconstruction from video for different applications. For example, animating the cloth of virtual characters could be guided by video, thus limiting manual intervention. Images were taken from [26] and [15]. 28

1.4 Examples of the surfaces to which we applied our methods. (a) Textured surface undergoing a simple deformation. (b) Methods relying on smoothness assumptions would typically fail reconstructing such sharp creases. (c,d) With much less textured surfaces, the shape of uniform parts must be infered from that of the textured ones. This requires deformation models that accurately represent the properties of the surfaces. 29

1.5 Using global representations would involve learning two different models for these surfaces, even though they are made of the same material. With local representations, we can use the same model for both cases, and simply need to combine them to form the correct global shapes. 31

3.1 Correspondences between a reference image and an input image. The point-to-point correspondences are shown as red lines going from the first image to the second one. 46

3.2 Obtaining image correspondences. A feature point is detected in the reference image, shown in the middle. Knowing the reference 3D shape of the mesh, on the left, and the camera projection matrix, we can retrieve the facet to which the feature point belongs, and define it in terms of its barycentric coordinates. The feature point can then be matched against points detected in the input image, shown on the right. This yields 3D-to-2D correspondences in terms of the unknown 3D mesh vertices in the input image. 47

3.3 Effective rank of matrix $\mathbf{M}$. (a) 88-vertices mesh seen from the same viewpoint as the one used for reconstruction. (b) Singular values of $\mathbf{M}$ for the mesh of (a). Note how the values drop down after the $2N_v = 176^{th}$ one, as predicted by the affine model of Section 3.2.2. The small graph on the right is a magnified version of the part of the graph containing the small singular values. The last one is zero up to the precision of the matlab routine used to compute it and the others are not very much larger. 51

3.4 Visualizing vectors associated to small singular values. (a) Reference mesh and mesh to which one the vectors has been added seen from the original viewpoint, in which they are almost indistinguishable. (b) The same two meshes seen from a different viewpoint. (c) The reference mesh modified by adding the vector associated to the zero singular value. Note that the resulting deformation corresponds to a global scaling. 52

3.5 Using silhouettes information. Texture edges, shown in red, are detected in the current image. The contour of the current deformed mesh is projected onto the image, here depicted in blue. We then look in the direction of the normal of this projected contour for a detected image edge. Finally, we minimize the distance between the image and projected contours. Note that edge detection yields wrong edges. Our algorithms must therefore be robust to such erroneous information. 56

4.1 Singular values for a 5 frames sequence under affine projection. Left: Without temporal consistency constraints between frames, the linear system has many zero singular values, which implies severe reconstruction ambiguities. Right: Constraining the z coordinates as discussed in Section 4.1.1 leaves the non zero singular values unchanged but increases the value of the others, thus removing the ambiguities. 60

4.2 Under a perspective camera model, the depth d of a vertex $\mathbf{v}$ at time $t+1$ can be obtained by projecting vector $\mathbf{v}^t\mathbf{v}^{t+1}$ on the line-of-sight. This is done through a dot-product between this vector and the normalized direction of the line-of-sight, which is a non-linear function of the vertex position. 61

4.3 Singular values for a 5 frames sequence under perspective projection. Left: Without temporal consistency constraints between frames, the linear system is ill-constrained. Right: Bounding the frame-to-frame displacements along the line of sight using the linear expression of Eq. 4.4 transforms the ill-conditioned linear system into a well-conditioned one. The smaller singular values have increased and are now clearly non-zero. Since our motion model introduces more equations than strictly necessary, the other values are also affected, but only very slightly. 63

4.4 Reconstructing an 88-vertices mesh using perfect correspondences that were corrupted using zero-mean Gaussian noise with variance five, which is much larger than what can be expected of automated matching techniques. Top: The original mesh and reconstructed one projected in the synthetic view used to create the correspondences. As expected, the projections match very closely. Bottom: The two meshes seen from a different viewpoint. 64

4.5 Distance between the original mesh and its reconstruction for each one of the 9 deformed versions of the mesh of Fig. 4.4. We plot five curves corresponding to vertex-to-surface distances obtained with variance one to five gaussian noise on the correspondences. The distances are expressed as percentages of the length of the mesh largest side. Note that knowing the final shape avoids having a monotonically increasing error. 64

4.6 Reconstructing a deforming sheet of a paper from a 250-frames sequence. Top: The reconstructed mesh is reprojected into the original images and closely matches the outline of the paper. Bottom: The same mesh seen from the side. In spite of local inaccuracies in depth, the overall shape is correct, which indicates that the ambiguities have been successfully resolved. 65

4.7 Reconstruction results for a plastic sheet, which is much more flexible than the sheet of paper of Fig. 4.6. In spite of this, the overall shape is again correctly recovered up to small errors due to erroneous correspondences. . . 65

4.8 A cone of radius γ is defined for each correspondence. It is centered in the camera, and its axis goes through the image measurement. 3D points on the surface must reproject inside their corresponding cone on the image plane. 67

4.9 Using a mesh imposes constraints on the reconstruction. (a) Without a mesh, nothing would prevent the 3D points to perfectly match their corresponding image locations. However, this would yield a completely meaningless shape. (b) The barycentric coordinates link several 3D points together, and thus impose a natural coherence between them. 68

4.10 The SOCP formulation of the correspondence problems is very sensitive to outliers, since it minimizes the L_∞-norm. (a) With a single outlier, the minimum cone radius γ remains very large, and thus allows the correct matches to reproject far from their corresponding image locations. (b) Once the outlier is removed, γ can take a much smaller value, which yields a much better shape. 69

4.11 Reconstructing a piece of paper using only the correspondences constraints of Section 4.4.1 but not the deformation constraints of Section 4.4.2. (a) The reprojection of the mesh is correct. (b) However, the 3D shape as seen from a side view is completely wrong because the depth ambiguities are not properly resolved. 70

4.12 We predict that the orientation of the edge between $\mathbf{v}_i$ and $\mathbf{v}_j$ at time $t+1$ will be the same as at time t. We then constrain the distance between vertex $\mathbf{v}_j^{t+1}$ and its prediction $\tilde{\mathbf{v}}_j^{t+1}$ to be less than some specified value. 71

4.13 Reconstructing an 88-vertices mesh with sharp folds using perfect correspondences that were corrupted using zero-mean Gaussian noise with variance two. The shape of the reconstructed mesh (blue) corresponds very closely to the original one (red). The meshes are seen from a different perspective than the one used to retrieve the shapes in order to highlight the differences. 72

4.14 We compare the results of our SOCP formulation (solid red) against those obtained using CFSQP, a constrained non-linear least-squares minimization (dashed blue) for the 50 frames of the synthetic sequence of Fig. 4.13, for noise variance $\sigma = 1$ and 2. In the top row, we show the distance between the original mesh and its reconstruction. Both methods give similar results but SOCP is about 50 times faster. In the second row, we give the median reprojection errors. For both methods, they are less than one pixel, even though CFSQP performs slightly better. Recall, however, that SOCP does not precisely minimize the reprojection errors, but enforces the reprojections to lie in a cone of a given radius. 73

4.15 Introducing non-convex inextensibility constraints, for noise variance $\sigma = 1$ and 2. Since CFSQP can handle such constraints, we introduce them into our CFSQP formulation and, as in Fig. 4.14, compare the results (in blue) against those of SOCP (in red). In addition to being much slower, CFSQP gives unstable results and fails to converge in some frames after 2000 iterations. 74

4.16 Influence of the number of correspondences in each facet on the reconstruction for noise variance $\sigma = 2$. We decreased the number of correspondences per facet from 10 to 1, and display the median (red line) and maximum (blue crosses) values of the same errors as in Fig. 4.14. We show the 3D distance errors in the left image, and the reprojection errors computed for all 10 correspondences per facet in the right one. Note that the 3D vertex-to-surface distance is little affected by the correspondences, whereas the reprojection error decreases in a more noticeable manner, which is to be expected, since, in the first cases, not all correspondences were used during optimization. 74

4.17 Reconstructing a deforming sheet of paper from a 71 frames video and a 116 frames video. The mesh is reprojected in the image in the top row and seen from a different perspective in the bottom one. Even though no smoothness constraint was enforced, the algorithm correctly recovered smooth deformations. 75

4.18 Reconstructing the deformations of a piece of paper with two sharp folds in it, so that that they are no longer smooth. Note that our method correctly recovers the creases. 76

4.19 Recovering the deformations of a plastic bag with a sharp crease in it from from an 86 frames video. 76

4.20 Recovering more complex deformations of the plastic bag. The first two rows depict the reprojection of the mesh into the original images and the mesh seen from a different perspective as before. In the third row, we overlay the mean curvature of the recovered surface on the images. The high curvature areas, shown in red, correspond to the actual creases that can be seen in the top row. In the fourth row, we overlay the level-lines of constant z on the images. We recommend viewing the last two rows in color as they might be difficult to interpret on a greyscale printed copy. . . . 77

4.21 Recovering the deformations of a piece of cloth from a 50 frames video. . . 77

4.22 Recovering the deformations of a piece of cloth with several folds. The third and fourth rows depict the same curvature and level-line information as in Fig. 4.20 and are best viewed in color. 78

4.23 Another example of a different deformation of that same cloth in a 61 frames sequence. The third and fourth rows depict the same curvature and level-line information as in Fig. 4.20 and are best viewed in color. 79

5.1 Hexagonal triangulations. (a) Rectangular mesh used to model a piece of paper. (b) Triangular mesh used to model a spinnaker. (c) Stitching a rectangular patch for the body part and two triangular ones for the sleeves lets us model a t-shirt. 82

5.2 Specifying the 3D shape of the rectangular mesh and subdvided triangle. (a) We fix the shape of the bottom row from left to right by rotating each facet with respect to its left neighbor. For each following row, we only need to set the angle between the leftmost facet and the one below and the angle between the rightmost facet and its left neighbor. (b) The angles between the facets of the bottom row are first set from left to right. For each upper row, only the angle of the first facet need be set. (c) Attaching two patches together. Because the base of each triangular patch is attached to the body, only one single angle is required to fully specify their first row. 83

5.3 Determining the position of interior vertices by the intersection of 3 spheres. The positions of solid lines triangles have already been computed. We seek to determine the position of point $\mathbf{P}$. This can be done by computing the intersection of 3 spheres of known radii centered in $\mathbf{C}_0$, $\mathbf{C}_1$, and $\mathbf{C}_2$, respectively. This yields between two and zero solutions depending on the configuration of the other triangles. 84

5.4 Deformation modes of the meshes of Fig. 5.1. In all figures, $\mathbf{y}_0$, the average mesh, is shown in red. The other two are obtained by taking a single mode weight to be non zero. A positive value of that weight yields the green mesh and a negative one the mesh shown in blue. Bending and extension modes of (a) the flat rectangular mesh, (b) the triangular spinnaker, and (c) the t-shirt. 85

5.5 Image data. (a) An image from an input sequence. (b) One of 15 images used to build a textured 3D model of the spinnaker. For our experiments, we added black scotch tape on the otherwise white parts of the sail to help our wide-baseline algorithm to find correspondences between model and input images such as those depicted by the black lines. (c) Contours detected as texture boundaries. Even though the boundary is not correct everywhere, thanks to the model and robust estimation we still recover the correct shape. 87

5.6 Fitting test surfaces created by varying the determining angles. Using 50 deformation modes proved sufficient to reconstruct the surfaces to a good precision. (a,c) The original shapes are shown as shaded. (b,d) The fitted ones are displayed as wireframes. 89

5.7 Convergence using synthetic data. (a) Projections of the synthetic surfaces used as input for the optimization process. (b) Examples of initializations. (c) Median of the mean distances between the vertices of the recovered mesh and the synthetic surface as a function of the number of correspondences that were used. The measures are given as a percentage of the longest side of the initial rectangle. We did not draw error bars because, as soon as we used more than 50 matches, the first and third quartile of the mean distances are indistinguishable from the median. 90

5.8 Deforming a sheet of paper. Top row: Deformed mesh projected on the original sequence as a wireframe. Bottom row: Deformed mesh shown as a wireframe model seen from a different viewpoint. Note that even the back deforms correctly. 91

5.9 Another deforming sheet. Top row: Projected wireframe. Bottom row: Deformed mesh shaded and seen from a different viewpoint. 91

5.10 Deforming fabric. The results are displayed in the same manner as in Fig 5.8. Since the fabric is highly textured, borders of the mesh are sometimes mismatched with texture edges, which results in small misalignments. 92

5.11 Tracking an inflating balloon with an extensible mesh. Note that the mesh keeps on covering the same portion of the balloon. In the last row, we re-textured the resulting mesh, and reprojected it into the images. 93

5.12 The inverse behavior as in Fig. 5.11 can be observed when the balloon deflates. 94

5.13 Tracking an inflating balloon with an inextensible mesh. The portion of the balloon covered by the mesh becomes smaller and flatter as the balloon expands. This also is a correct solution if we do not assume that the mesh represents a particular portion of the object, but rather models the behavior of a fixed size part, which is valid when relying on image-to-image matches only. 95

5.14 Superposition of the initial mesh in red and the final one in blue. Left: When extension is not penalized, the mesh increases as the balloon inflates. Right: When we enforce the edges to remain of constant length, the surface remains of same area, but becomes flatter. 95

5.15 3D model of the spinnaker overlaid on the three images used to compute its reference shape and texture. In each image, we specified 10 correspondences with a CAD model of the spinnaker and used them, along with automatically detected silhouettes, to deform it. We assume that the spinnaker did not deform in these images because they were taken in quick succession by a chase-boat. 96

5.16 Tracking a deforming sheet of paper and a t-shirt. In both cases, we show the deformed 3D mesh overlaid on the original images in the top row and then seen from a different viewpoint in the bottom row. 96

5.17 Tracking a spinnaker with either one or two cameras. (a,b) Two synchronized images from independently moving cameras, with recovered spinnaker reprojection. (c) Tracking using only one camera. Note that, once reprojected on the images, the results are almost indistinguishable. (d) 3D results with two cameras. Both camera positions are also retrieved. (e) Superposed 3D shapes retrieved using either one (red) or two (blue) cameras. Note that both shapes are very similar, which indicates that the deformation model provides a good approximation when data is missing. 97

5.18 Tracking an extensible surface undergoing anisotropic deformations. In the top row, we show the original images and, in the bottom row, we overlay the recovered 3D grid that stretches appropriately. 97

5.19 Evaluating the accuracy of our approach. (a,b,c): Images from videos acquired using three synchronized and calibrated cameras. Image (b) belongs to the video we used to monocularly reconstruct the 3D shape using our method and, then, re-projecting it into the image. (d) We triangulated the 3D coordinates of the 10 keypoints shown as crosses by manually establishing correspondences in images (a) and (c). (e) We repeated this operation every 10 frames and plot the average differences between the x-, y-, and z-coordinates of those manually computed and those derived from our automated and monocular reconstruction. (f) We also computed the Euclidean distances between the monocular reconstructions and the manually computed points, and plot their medians together with values at 25% and 75%. 98

5.20 (a,b) Original and side views of a surface used to generate a synthetic sequence. The 3D shape was reconstructed by an optical motion capture system. (c,d) Eigenvalues of the linear system written from correspondences randomly established for the synthetic shape of (a). (c) The system was written in terms of 243 vertex coordinates. One third of the eigenvalues are close to zero. (d) The system was written in terms of 50 PCA modes. There are still a number of near zero eigenvalues. (e) First derivative of the curve (d) (in reversed x-direction). We take the maximum value of N_l to be the one with maximum derivative, which corresponds to the jump in (d). 100

5.21 Lin-Log plot of the number of equations and number of unknowns as a function of the number of Extended Linarization iterations for the case of a 10×10 square mesh. Note that after 4 iterations, the number of equations exceeds the number of variables, making the system, in theory, solvable. However, the size of the system is of the order 10^{12}, which makes its solution intractable in practice. 102

5.22 Shape recovery of a 200×200mm synthetic mesh imaged by a virtual camera placed 20cm away from it. Each plot shows the mean vertex-to-vertex 3D distance between the recovered surface and the ground-truth as a function of its mean curvature. The three different curves in each graph correspond to a varying number of correspondences per facet. Left to right, the number of outliers grows. Top to bottom, the gaussian noise added to the correspondences increases. For each experiments, we plot the average over 40 trials. The last row shows in blue recovered shapes for the ground-truth surface of Fig. 5.20(a,b), shown in red. The corresponding mean vertex-to-vertex distances are 9mm, 19mm and 38mm. This highlights the fact that even for distances around 40mm, the recovered shape remains meaningful. . 105

5.23 Comparison of our closed-form results against the results of constrained optimization. Optimization was performed on the vertex coordinates using Matlab's `fmincon` function, and starting from the flat position. (a) Mean vertex-to-vertex distance. (b) Reprojection error. Constrained optimization is both much slower and far less accurate than our approach. 106

5.24 3D registration of a folded bed-sheet to an individual image given a reference configuration. Top row: Recovered mesh overlaid on the original image. Middle row: Synthesized textured view using the recovered shape. Bottom row: Real side view of the sheet from similar viewpoints. Despite lighting changes, the synthetic images closely match the real ones. 107

5.25 Shape recovery of a bed-sheet. Top row: Recovered mesh overlaid on the original image. Bottom row: Mesh seen from a different viewpoint. 108

5.26 Shape recovery of a piece of cloth. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 108

5.27 Shape recovery of the central part of a t-shirt. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 109

5.28 In such cases where a large part of the drawing on the t-shirt is hidden, or where the image becomes too blurry, not enough feature points could be found. We therefore fixed a threshold, and only recovered the shape in images where we found a least 30 SIFT correspondences. Note that, since we are not tracking the surface, this does not prevent us from correctly recover the shape in the other frames. 110

6.1 Advantages of the local models over global ones. (a) Highly flexible surfaces may undergo too complex deformations for the global models, whereas, locally, these deformations remain relatively simple. (b) A new global model must be learned for this surface even though it is made of the same material as the surface in (a). Local models can be combined into surfaces of arbitrary shapes. 112

6.2 We used an optical motion capture system to acquire real training data. Left: We stuck reflective markers as a rectangular grid on the surface of interest. Right: We deformed the object in front of six infrared cameras. . . 116

6.3 Decomposing the surface into patches. In this case, the global surface $\tilde{\mathbf{y}}$ is composed of four overlapping patches $\mathbf{y}_{1,..,4}$. 117

6.4 Validating the linear deformation models. We compute the mean of the average vertex-to-vertex distances between test data and the model predictions, and plot it versus the number of modes. Top: The error for the cardboard model (left) decreases faster than for the napkin one (right). This corresponds to our intuition that fewer dimensions are necessary to model the deformations of a more rigid material. Bottom: We tried to reconstruct the napkin data using the cardboard model (left), as well as the opposite (right). We can observe that, since the napkin is more flexible, its deformations are a superset of those of the cardboard, and thus can model it quite accurately. This is not the case when trying to reconstruct napkin data with the cardboard model. 121

6.5 Validating the non-linear deformation models. The same reconstruction error as in the linear case is computed and plotted as a function of the latent dimension. Top: In the cardboard case (left), we chose latent dimension 4, since it corresponds to the point where the error stabilizes. In the case of the napkin (right), we chose to use dimension 7 because of convergence problems during training in dimensions 8 and 9. These would have required a larger number of inducing variables, which would have incerased the computational burden. Bottom: As in the linear case, we can observe that the cardboard models are unable to reconstruct napkin data, whereas the inverse is possible. 123

6.6 We used Isomap to compute the low-dimensional embeddings of our cardboard (left) and napkin (right) data for different latent dimensions. We plot the residual variances given by Isomap as a function of the dimension. This confirms our choice of dimension 4 for the cardboard and 7 for the napkin. . 124

6.7 Synthetic images generated from optical motion capture data (a) Shaded view of a cardboard surface (b) Similar shaded view for a paper napkin. (c,d) Images synthesized by texture-mapping using either a rich texture or a much more uniform one. 124

6.8 Comparison of the linear (dashed red) and non-linear (solid blue) models for cardboard using sequences of synthetic images. No penalization of extension was used to obtain these results. Top row: For each of the well-textured images, we plot, on the left, the mean 3D vertex-to-vertex distance, and, on the right, the mean reprojection error of randomly sampled surface points. Bottom row: Same plots for much less textured images. Note that the non-linear models yield a better 3D reconstruction than the linear ones. This is to be expected since they suffer less from the absence of stretching penalty. 125

6.9 Same plots as in Fig. 6.8 for the napkin models. This time, the linear models perform as well as the non-linear ones. This can be explained by the fact that the deformations of a flexible material remain closer to the flat shape than that of a more rigid one. Thus, even when stretching, the linear models yield reasonable 3D errors, but visually less accurate shapes, as can be checked from the last row of the figure which depicts the linear (left) and non-linear (right) reconstructions of the same ground-truth (middle) corresponding to frame 60 of the less-textured sequence. 126

6.10 3D reconstruction errors for the linear and non-linear models when using inextensibility constraints. Top: Mean vertex-to-vertex distance as a function of time for the cardboard textured (left) and less-textured (right) sequences. Bottom: Same plots for the napkin sequences. Note that in both cases, the linear models strongly benefit from the inextensibility constraints, whereas the non-linear ones are less affected. 127

6.11 Reconstructing a rectangular piece of cardboard from a single video. In each of the three examples, we show the recovered surface overlaid in red on the original images, and the surface seen from a different viewpoint. As shown in the top rows, a complete absence of texture leads us to retrieve a surface that is plausible, but not necessary accurate. It is only one of a whole family of equally likely solutions. However, this problem is fixed by adding very little image information, as shown in the other two examples. We then recover deformations that match the real ones. 128

6.12 Reconstructing a rectangular piece of cardboard from a single video with linear local models. In the first row, we show the surface recovered with inextensibility constraints overlaid in red on the original images. In the middle, we show a side of the surface recovered without penalizing stretching. In the bottom row, we can see that inextensibility constraints improve the 3D shape, and therefore should be used in conjunction with the linear local models. Note that we do not show the reprojection of the compressed surface, since it yields similar images as in the first row. 129
6.13 Reconstructing a circular piece of cardboard with the same local models. Note that assembling square patches only allows us to approximate the object's outline. This prevents us from using image edges, but does not stop us from successfully recovering the deformations. In the middle row, we show the results of the non-linear models, and in the bottom row, those of the linear ones. 129
6.14 Despite a very large occlusion, we manage to reconstruct a deforming piece of cardboard in each frame of a sequence. Note that even if some small reconstrution errors occur, the global shape nevertheless matches the true one. 130
6.15 Reconstructing a much more flexible paper napkin. As opposed to cardboard, results obtained with non-linear models (first and second rows) are better than with linear ones (third and fourth rows). This confirms our intuition that complex deformations are non-linear. 130
6.16 Reconstructing a different deformation of the same napkin. Even though there is little texture, the 3D shape of the surface is correctly recovered, as shown in the bottom row where the surface is seen from a different perspective. 131
6.17 Reconstructing a napkin of different topology with non-linear (top) and linear (bottom) models. As with cardboard, assembling square patches only allows us to approximate the outline of the hole, but still lets us recover correct deformations. 131
6.18 Modeling the napkin without explicitly accounting for the hole. The local models are replicated to cover the whole rectangular surface. Note that the surface does not always reproject correctly, as can be seen at the hole upper boundary. The hole creates discontinuities in the surface, which modifies the global behavior. It should therefore be modeled explicitly. 132
6.19 Using local models to track the same extensible surface as in the global case. In the top row, we show the original images, the second row displays our results with a global model, and in the bottom row, we show the surfaces obtained with local models. Note that the improvement in texture matching most probably comes from using template matching rather than correspondences. Furthermore, the change of mesh resolution was only introduced for convenience of use with the local models. 133

6.20 Mean vertex-to-vertex distance (left) and mean reprojection error (right) for the closed-form reconstruction using linear local models. The dashed red curve corresponds to the closed-form solution, and the solid blue one to the refined solution after optimization. Top row: The errors were computed for the textured cardboard sequence from correspondences synthetically sampled on the facets of the mesh. Bottom row: Same plots for the textured paper napkin sequence. 135

6.21 Similar plots as in Fig. 6.20, but with more realistic matches obtained with SIFT [103]. Left: We plot the errors obtained with synthetic matches (dashed red) and with SIFT matches (solid blue) as a function of time for the cardboard case. Right: Same plots for the paper napkin sequence. . . . 136

6.22 We studied the influence of the number of modes by computing the solutions with 20 modes rather than the original 75 ones. Left: We show the errors obtained from synthetic matches with 75 modes (dashed red) and 20 modes (solid blue), and from SIFT matches with 20 modes (dotted black) for the cardboard case. Right: Same plots for the paper napkin sequence. . . 136

6.23 Recovering the shape of a piece of paper. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 137

6.24 Recovering more complex deformations of a piece of paper. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 138

6.25 Reconstructing a sharp fold in a piece of cloth. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 138

6.26 Recovering the shape of a plastic bag. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations. 139

7.1 Effect of adding (green) or subtracting (blue) a single deformation mode to the rest shape (red). From top to bottom, modes were obtained with real cardboard data, real napkin data, synthetically generated meshes of two different topologies, a stiffness matrix for cardboard, and a stiffness matrix for the napkin. The plots display the first five modes other than global motion. 142

7.2 Comparison of the reconstruction error as a function of the number of modes for the different linear global models. On the left, the error was computed on the cardboard validation data, whereas on the right it was done for the napkin case. From top to bottom, we display such errors for the cardboard model, the napkin model, the model obtained from synthetic data, and the physics-based models. 143

7.3 Residual variance obtained with Isomap on the cardboard (left) and napkin (right) global training data. These plots suggest latent dimensions 16 for the cardboard and 30 for the napkin. 144

7.4 Reconstruction errors on synthetic images generated from validation data. We applied a nicely textured image (left) or a much more uniform one (right) to ground-truth meshes, and projected them with known camera. The top row corresponds to a piece of cardboard, and the bottom one to a paper napkin. In each plot, we show the results for the global models obtained from real data (red), synthetic data (blue), and modal analysis (black). Note that the latter performs significantly worse as the others in the napkin case. 145

7.5 Visual interpretation of the local deformation modes. From top to bottom, we show the cardboard, napkin, synthetic, cardboard stiffness, and napkin stiffness modes. Note that they closely ressemble the global ones. 146

7.6 We computed the mean predicitions of the non-linear local models for the minimum (green) and maximum (blue) values for a single latent dimension while setting the others to zero. Here, we show the deformations along the 4 possible dimensions for our cardboard model. 147

7.7 Same plot as in Fig. 7.6, but for the 7 dimensions of the paper napkin model. 147

7.8 Comparison of the different local models. We plot the average reconstruction error on cardboard (left) and napkin (right) test patches. From top to bottom, the models used were obtained from cardboard data, napkin data, synthetic data, and with modal analysis. 149

7.9 Learning a non-linear model from synthetic data. (a) Running Isomap on the data gives a residual error that strongly decreases until dimension 4, but that truly stabilizes at dimension 10, as shown in the zoomed-in plot (b). (c) We learned non-linear models for latent dimensions up to 10 and tested them on real cardboard and napkin data. These models achieve poor performance compared to those trained on real data. 150

7.10 Reconstruction errors on synthetic images generated from validation data. We applied a nicely textured image (left) or a much more uniform one (right) to ground-truth meshes, and projected them with known camera. The top row corresponds to the piece of cardboard, and the bottom one to the paper napkin. In each plot, we show the results for the local models obtained from real data (red), synthetic data (blue), and modal analysis (black). 150

7.11 Reconstruction errors on synthetic well-textured cardboard images using all the methods discussed in this thesis. 151

7.12 Same plots as those of Fig. 7.11, but for much less textured cardboard images. Since some methods rely on correspondences, we could not apply them to this case. 152

7.13 Reconstruction errors on synthetic well-textured napkin images. We compare our methods together and with smoothness and inextensibility constraints. 153
7.14 Same plots as in Fig. 7.13, but for much less textured napkin images. Since some methods rely on correspondences, we could not apply them to this case. 154
7.15 We compared our methods on a real video of a deforming sheet of cardboard. In order to texture it, and thus make it usable for correspondences-based techniques, we stuck a piece of paper on the cardboard. Since cardboard is more rigid, its behavior remains unchanged. Top row: Reprojection of the results obtained with the linear local model. Next rows: Results obtained, from top to bottom, with linear motion model, convex optimization, linear global model by tracking and in closed-form, non-linear local models, linear local models by tracking and in closed form. 155
7.16 Same comparison as in the cardboard case but for the more flexible napkin. In this case, we computed correspondences from the results obtained with the non-linear local models. 156
7.17 Comparing our approaches on a piece of paper with sharp creases. From top to bottom, we show the results of the convex optimization approach and the closed-form solutions of global and local models. Methods relying on a deformation model are not truly adapted to this case, and tend to oversmooth the surface. Nonetheless their results are reasonable approximations of the true shapes. 157
7.18 We compared the reconstructions with local (top) and global (bottom) models of a surface with a hole. Even though the global model manages to track the surface, it oversmoothes the parts where the hole creates discontinuities. This yields a bad reprojection of the resulting mesh. 158
7.19 Comparison of the local and global models on a piece of cardboard from which a large part was cut out. This simulates the extreme case of a rectangular hole where the two opposite parts can move independently. In the top two rows, we show the results of the local models that correctly recover the shape. In the middle, we display results obtained with a global model for which we penalized stretching. For small deformations, this representation still works, but the two parts cannot be moved far apart, since this would violate inextensibility constraints. Without these constraints, the mesh reprojects correctly on the images, but gives a meaningless 3D shape, as shown in the bottom two rows. 159

8.1 Some of our techniques have been used in a software that computes the shape of the spinnaker of the swiss sailing team Alinghi. 162
8.2 Similarly as for the spinnaker, our methods were applied to recover the shape of the main sail of the french boat Hydroptère. 163
8.3 Images of the french boat Hydroptère [71]. 164

1 Introduction

Deformable surface 3D reconstruction from a single viewpoint is an active area of research in the Computer Vision community. Whereas it seems easy for a human being to see the 3D shape of an object, it becomes a challenging and ambiguous problem for computer-based techniques. This is especially true when the sensor data is noisy, which is typically the case when dealing with real images.

The goal of this thesis is therefore to study the ambiguities inherent to monocular 3D deformable surface reconstruction and to propose ways of overcoming them. The common thread in this work is that real surfaces do not deform randomly and cannot assume completely irrational shapes. As a consequence, one may introduce some knowledge of what is feasible and what is not to constrain the recovery and remove the ambiguities. In the various methods we present here, such knowledge ranges from generic assumptions that hold for most surfaces with potentially different physical properties to much more precise ones that closely correspond to a specific material.

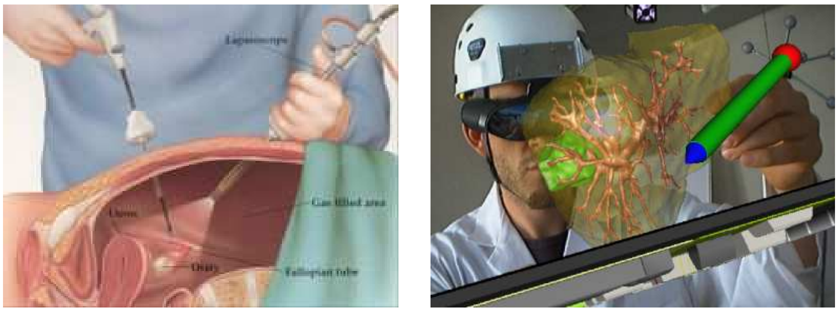
In the remainder of this chapter, we first describe in more details the problem we address and some possible applications of the solutions we propose. We then summarize the contributions of this thesis.

1.1 Problem Definition

In this work, we seek to recover the 3D shape of a non-rigid surface given an image, or a sequence of images of the surface deforming in front of a single camera. Furthermore, we assume that we have a reference image of the object of interest for which we know the 3D shape. In practice, we may, or may not assume that the camera is fully calibrated and remains fixed. Similarly, the reference image may, or may not have been taken from the same viewpoint or be the first image of the video sequence. Note that requiring a reference image, even taken from a different angle, does not mean that we can use standard stereovision techniques since the shape of the object is not the same in both images. As will be explained in more details in Section 3.3, the reference image is used to extract the necessary texture information from which we can retrieve a 3D shape, but not triangulation with the input image.

Apart from being a fascinating problem, non-rigid 3D shape recovery has applications in many different domains:

- A first example is medical imaging. For the patient's well-being, surgery tends to become less and less invasive. This implies smaller and smaller cuts in the patient's skin, which do not give the surgeons a direct view of their work. They only leave



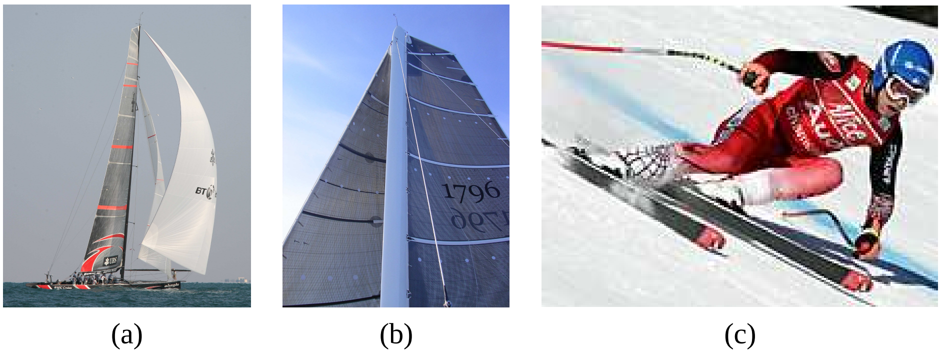
(a) (b)

Figure 1.1: Surface reconstruction can be applied to medical imaging. (a) Schematic representation of non-invasive surgery (b) Augmenting a scene with the reconstruction of a virtual liver [130].

enough space for small cameras to be introduced into the patient's body. In such conditions, the resulting images are of poor quality, and make the surgeons' work much harder. As shown in Fig. 1.1, having a full 3D representation of the organ's surface recovered from the images, or an augmented view of the organs, would certainly help them greatly.

- Many sports could benefit from a system that reconstructs non-rigid 3D shapes from video. For example, as shown in Fig. 1.2 (a,b), sailors want to analyze the effect of their maneuvers on the shape of their sails, or, sometimes even more interestingly, study the sails of their opponents. In this context, video presents a clear advantage on other sensors that should be placed on the sail itself, thus changing its behavior. Similarly, one might want to analyze how skis deform during a race to improve their design, as illustrated in Fig. 1.2 (c).
- The entertainment industry could benefit greatly from improved techniques for video-based shape recovery. In animation movies, video games, or special effects, many things are still done manually, image after image. A lot of time could be saved if the deformations of the clothes of animated characters, such as those of Fig. 1.3, could simply be obtained by filming a real person performing some motion, reconstructing his or her clothes in 3D, and re-applying the resulting deformations to the animated character. Similarly, as Augmented Reality becomes increasingly popular, it will become increasingly important to accurately model the environment and its deformations in order to correctly augment the scene with virtual objects. For example, this could be used to draw virtual advertisment logos on sportsmen's clothes, or boat sails, thus avoiding the need to physically print them and making easy to change them as necessity dictates.

Unfortunately, recovering the 3D shape of a surface from a single view is an ill-constrained problem. The high number of parameters and the noisy image information make it imprac-



(a) (b) (c)

Figure 1.2: Many sports could benefit from 3D shape recovery. (a,b) Sailors are interested in knowing the shape of their own sails, or of the opponents' sails (c) Recovering the shape of skis during a race could help improving their design.

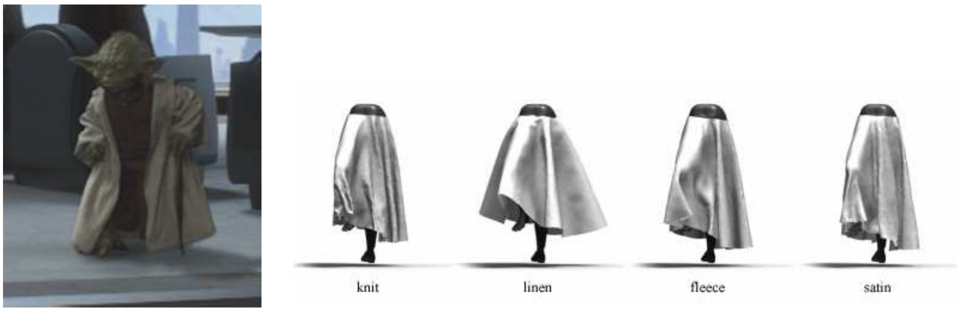
tical to solve without prior knowledge of the possible deformations that the surface can undergo. All the above-mentioned applications therefore require the use of deformation models, such as those we study in this thesis.

1.2 Contributions

In this work, we focused on the reconstruction of deformable 3D surfaces from video sequences. As mentioned above, we assume that we are given an image of the surface in which its shape is known and that we can establish point-to-point correspondences between that reference image and image sequences in which the surface deforms. Our goal is to propose algorithms that are as generic as possible to reconstruct the deforming 3D shape from such data and from additional cues such as surface contours, when available.

A necessary first step, which has been neglected in earlier works, is to study the ambiguities of the shape reconstruction problem in this context. To this end, we represent the surface as a triangulated 3D mesh and use projective geometry techniques to show that recovering its shape and motion from point correspondences amounts to solving a linear system whose unknowns are the 3D coordinates of the mesh vertices. The ambiguities stem from the fact that this linear system is very ill-conditioned and that a whole subspace of shapes can be considered as solutions. Additional knowledge and constraints are therefore required to pick the right one.

We propose different ways of imposing such constraints. They range from very generic ones that make minimal assumptions on the surface physical properties to much more specific ones that rely on learning these physical properties. The former are very generic but only applicable when the image data provides information over the whole surface, which, in practice, requires the surface to be very textured, as in Fig. 1.4(a,b), since we rely on point correspondences. By contrast, the latter can operate effectively on surfaces that are far less textured, as depicted in Fig. 1.4(c,d), but require data from which we infer the



(a) (b)

Figure 1.3: The entertainment industry could use 3D reconstruction from video for different applications. For example, animating the cloth of virtual characters could be guided by video, thus limiting manual intervention. Images were taken from [26] and [15].

objects' properties. The situations for which the various models we have proposed are effective are summarized in Table 1.1.

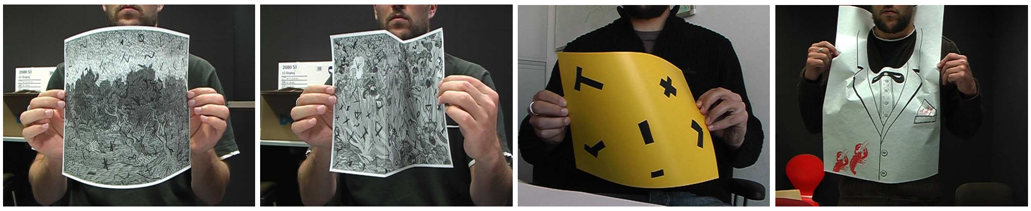
Whether dealing with very textured deformable surfaces or relatively untextured ones, our contribution is to completely formalize the 3D shape and motion recovery problem and to derive practical algorithms from this formalization. This is important because most real-world surfaces combine textured and uniform parts, and practical solutions will have to deal with this state of affairs. Conceivably, the algorithms that will eventually be deployed will reconstruct the most textured parts of surfaces, learn their physical properties from these reconstructions, and use them to model the less textured parts.

In the remainder of this section, we briefly introduce the different ways to impose the necessary constraints to overcome the ambiguities inherent to 3D shape and motion recovery from single videos. They will be discussed in more details in the following chapters.

1.2.1 Motion Models

Given a surface of unknown physical properties, one of the most reasonable models is to relate the deformation in the current frame to that in the previous ones. The resulting dynamical model is, of course, very rarely exactly correct. However, for optimization purposes, when one seeks to minimize some error defined as an objective function, such a simple motion model can prove surprisingly effective. It relieves the need to enforce surface smoothness, as is done in many current methods, and thus allows the recovery of complex and possibly discontinuous deformations, such as those that result in sharp folds and creases, as shown in Fig. 1.4(b).

More formally, in this work, we study two different motion models. The first one simply seeks to minimize the frame-to-frame motion in depth of the mesh vertices to avoid implausible configurations. The second one involves formulating the reconstruction as a Second Order Cone Programming (SOCP) problem solved using standard convex op-



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

Figure 1.4: Examples of the surfaces to which we applied our methods. (a) Textured surface undergoing a simple deformation. (b) Methods relying on smoothness assumptions would typically fail reconstructing such sharp creases. (c,d) With much less textured surfaces, the shape of uniform parts must be infered from that of the textured ones. This requires deformation models that accurately represent the properties of the surfaces.

timization techniques [21]. This ensures that the orientation of the mesh edges will not change drastically between two consecutive frames of the video sequence.

1.2.2 Global Models

While very generic, the approach described above suffers from several limitations. First, it provides no way to infer the shape of untextured parts of the surface. Second, penalizing image-to-image motion is only practical if one knows the shape in at least one frame of the sequence, and does not prevent drift when one tracks from frame to frame. When the surface's physical properties can be modeled, these weaknesses can be overcome by replacing frame-to-frame constraints by regularization ones within individual frames. We will show that this not only gives us the ability to reconstruct the shape of the whole surface, including untextured portions of it as shown in Fig. 1.4(c,d), but also allows us to do this in closed-form in single images. In other words, given a set of correspondences, we can compute the shape without any prior estimate. This is a critical ability if surface reconstruction techniques are ever to be incorporated into working systems that must be able to initialize and reinitialize themselves automatically.

Since accurately modeling the physics of a surface undergoing large deformations is extremely complex, an effective way to derive appropriate regularization constraints is to learn deformation priors from training examples. Given a set of deformed 3D shapes, we seek to approximate their statistical distribution. This can typically be done in a space of reduced dimension since the degrees of freedom of the mesh are usually coupled, which makes the task easier.

The bottleneck in putting this approach into practice is the availability of databases of representative shape deformations, which must be quite large since representing deformable surfaces as triangulated meshes involves many degrees of freedom. In some cases, the databases can be obtained by simulation, but not always. To overcome this limitation, we introduce a novel approach to automatically synthesizing such databases for

	Well-Textured Surfaces	Poorly-Textured Surfaces	Frame-to-frame Tracking	Closed-Form Solution	Shape Indep.	Sharp Creases
Linear Motion Models	+	-	+	-	+	+
SOCP Motion Models	+	-	+	-	+	+
Linear Global Models	+	+	+	+	-	-
Non-Linear Local Models	+	+	+	-	+	-
Linear Local Models	+	+	+	+	+	-

Table 1.1: Summary of what the different models presented in this thesis can and cannot handle.

inextensible surfaces. The idea is that the shape of an inextensible triangulated mesh can be fully determined by a subset of the angles between its facets. Shapes are created by randomly setting these angles, and are therefore directly registered to one another. We then apply Principal Component Analysis (PCA) [81], a linear dimensionality reduction technique, to our database, and use the resulting deformation modes to recover the shape of surfaces from images.

For this purpose, we first propose a simple least-squares optimization method that recovers the shape in a video sequence by initializing the shape in the current frame with that of the previous one. However, this again means tracking the deformations from image to image, and thus suffers from error accumulation, as before. Therefore, to avoid this issue, we propose a closed-form solution to 3D reconstruction from individual images. As discussed at the beginning of this section, it starts from the fact that, given a set of correspondences between a reference image in which the shape is known and an input image in which it is not, the set of possible shapes lies within a linear subspace and can be expressed as a linear combination of vectors forming an orthonormal basis of that subspace. Assuming that the surface is inextensible then implies that the edges of the mesh that represents it retain their length, which translates into a set of quadratic equations on the coefficients of the linear combination. This set of equations can be solved using techniques such as Extended Linearization [40], thus providing us with the desired result.

1.2.3 Local Models

The approach described above is very effective when the goal is to model a specific surface for which a model can be created and stored. However, this model cannot be directly applied to any other surface whose rest shape is different, even if it is made of the same material. This, nonetheless, can be solved by replacing the global models described above by local deformation models that describe the behavior of small surface patches, which

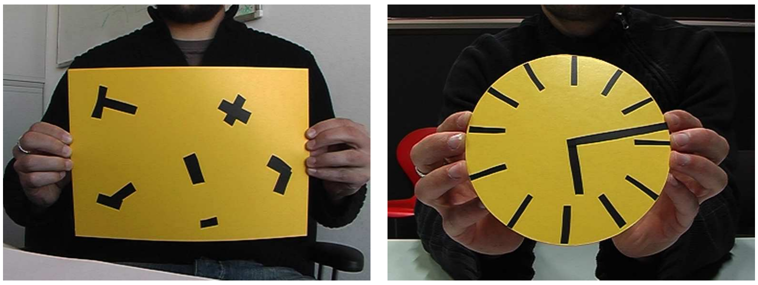


Figure 1.5: Using global representations would involve learning two different models for these surfaces, even though they are made of the same material. With local representations, we can use the same model for both cases, and simply need to combine them to form the correct global shapes.

can be combined into global models using a Product of Experts approach [64]. Not only does this yield a single model per material, as opposed to one per object shape, as shown in Fig. 1.5, but it also reduces the necessary amount of training data, since small portions of a surface can only undergo much simpler deformations than the whole surface itself.

Given training data obtained using an optical motion capture system, we learn local surface deformations models as Gaussian Process Latent Variable Models [94]. This produces a low-dimensional latent representation of the local shape manifold together with a mapping from this low-dimensional space to the high-dimensional one. This mapping is entirely defined by a covariance matrix, which can be taken as a linear or non-linear function of the latent variables. To represent more accurately the manifold of local deformations, we first learned non-linear models. While the global models obtained by combining these non-linear local models proved very effective at tracking deformations from frame to frame, they were not suited for surface reconstruction from individual images as before. We therefore turned to the simpler linear models that represent local deformations as linear combinations of deformations modes. Even though the resulting global models were slightly less accurate, we found the linear approach to have two key advantages. First, the local models are much easier to build and can be created even in the absence of training data using the technique introduced in Section 5.1.2. Second, and more importantly, they still allow for closed-form computation of the shape using the same approach as before.

1.3 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 describes the related approaches found in the litterature. In Chapter 3, we formalize the reconstruction problem and exhibit its ambiguities. In Chapter 4, we describe how to remove the ambiguities by introducing frame-to-frame constraints in the reconstruction, first as linear constraints, and then in a convex optimization framework. In Chapter 5, we present our global deformation models and our method to automatically create the required training data. We show how to use them both for frame-to-frame tracking and for reconstruction from individual images. Chapter 6 introduces our local models together with the corresponding shape recovery framework. We then compare the various proposed methods in Chapter 7, and conclude in Chapter 8 by summarizing our results and giving different future directions of research.

2 Related Work

Modeling the behavior of non-rigid surfaces has been an active area of research for the past twenty years. Many approaches have been proposed in the context of both Computer Vision and Computer Graphics. These two fields are closely related, since Computer Vision aims at solving the inverse problem of Computer Graphics, that is recovering an object's shape as opposed to simulating an object's deformations. It is therefore not surprising that similar representations often appear in both domains.

The most popular geometry-oriented techniques can be roughly classified into physics-based methods, statistical learning based approaches, and alternative shape representations. While different from our approaches, these methods are strongly related to our own work. As we will explain in more details in the remainder of this chapter, they suffer from weaknesses that we propose to overcome. However, their strengths inspired our work, which makes them worth discussing in this thesis.

In addition to shape constraints, 3D reconstruction can rely on various sources of image information, such as using multiple views, or shading. Even though in our own work we mostly rely on texture and edges obtained from a single viewpoint, such information could potentially be added to our algorithms to make them more robust. The goal of this work is to study shape recovery from minimal inputs. However, in practice, one should always rely on all the available sources of information. We therefore review the most relevant works on these topics as they ultimately would be included in a working system. Note that these methods could also benefit from our models, since they require conditions that are rarely satisfied in real life.

2.1 Physics-Based Methods

The initial approach to modeling deformations of non-rigid objects was inspired by mechanical engineering. The key idea is to model the behavior of an object by describing the true physical laws that govern it. The pioneering work in this field [84] was introduced to delineate 2D shapes in images, and was quickly extended to 3D [151, 152]. In this formalism, a global energy, built as the difference between an internal energy and an external one, is minimized. The internal energy decribes the physical properties of the object, and is typically separated into a bending term and a stretching term. The external energy corresponds to the image information. Image features act as external forces that deform the surface of interest.

2.1.1 The Finite Element Method

The formulation presented by Terzopoulos and his colleagues closely follows the standard mechanical engineering procedures, where one seeks to study the deformations of structures such as beams, plates, shells, or full 3D bodies. The usual approach to tackle this problem is to solve it via the Finite Element Method (FEM) [12, 183]. The structure of interest is then modeled as a discrete set of elements, such as beams, triangles, or tetrahedra, that are linked by their nodes. Following the laws of mechanics, mass, stiffness, and damping matrices are built for each element, and assembled to model the whole structure. One then seeks to solve the system of differential equations

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \ , \tag{2.1}$$

where $\mathbf{u}$ is the unknown displacement of the nodes, $\mathbf{M}$, $\mathbf{D}$, and $\mathbf{K}$ are the mass, damping, and stiffness matrices respectively, and $\mathbf{f}$ is the external forces. This models a full dynamical behavior, and can be simplified to the static case by neglecting the displacement derivatives. The matrices typically depend on material parameters, such as Young's modulus, Poisson's ratio, shear modulus, and thickness of the structure.

When considering small deformations, that is bearly visible deformations, of a materially linear object, the matrices in the previous equation remain constant and the system can be solved directly. However, the deformations have to be much larger to be observable in images. The problem is then said to be geometrically non-linear. Additionally, the material which the object is made of may exhibit a non-linear behavior, such as hyper-elasticiy, or plasticity. In the presence of either of these two types of non-linearities, geometric or material, the stiffness matrix becomes a function of the displacements, and the whole problem becomes much more complex. Computing a solution is then very expensive and often unstable due to phenomena such as buckling or critical points that yield different solutions.

In the non-linear case, several resolution methods have been studied. First, the Total Lagrangian approach, where the solution is computed starting from a reference configuration that will remain unchanged throughout the computation. Second, the Updated Lagrangian approach, where the solution is computed in several iterations for which the reference configuration is replaced by the current solution. Finally, the corotational approach, where a large deformation is separated into rotations of the elements and small deformations. Nowadays, the Updated Lagrangian and Corotational approaches are the most commonly used.

2.1.2 Physics-based Methods for Computer Graphics

Physics-based methods quickly became very popular in Computer Graphics. An extremely important area of interest is the modeling of clothes [69] because animated virtual characters typically wear garments that deform as they move. In the absence of good deformation models, artists must manually design their shape in each frame of a sequence. Physics-based models therefore constrain the feasible deformations of clothing, and make animation much easier. Several cloth models have been proposed, ranging from early versions [167, 119] that only achieved visually plausible results to much more accurate and realistic models [26, 27].

Since the physics-based approach typically yields computationally expensive algorithms, there has been a number of attempts at improving the resolution of such problems. More specifically, a classical issue was that very small time steps had to be taken to avoid numerical instabilities. Implicit time integration was therefore introduced to overcome this issue [8]. A comparison of other approaches can be found in [168]. Other techniques have been proposed to speed up the simulation process, such as the use of the Boundary Element Method [79], an alternative to FEM where the original differential equations are replaced by integral equations over the boundary of the object.

Similarly as advances in mechanical engineering led to improved computer graphics methods, subdivision surfaces [30, 45], already well-known in the graphics community, were introduced to the mechanical engineering community in the context of finite elements. They involve representing a surface with a coarse mesh, which can then be refined following a subdivision scheme [102]. This reduced the complexity of the finite element models, thus yielding more efficient representations [33].

Finally, accurate non-linear FEM was also studied in Computer Graphics for surgery simulation purposes [126], and for general deformable objects modeling [65, 175, 9]. The corotational approach proved succesful in this context of large deformations [112, 61], as well as other representations such as discrete shells [56], or invertible finite elements [78]. Accurate non-linear representations being very complex, simplifications have been proposed to yield physically plausible deformations based on elastically coupled rigid cells [20].

2.1.3 The Computer Vision Approach

At the same time as they were introduced in Computer Graphics for simulation and animation purposes, the physics-based models became highly popular in Computer Vision for non-rigid motion analysis [83]. As in Computer Graphics, the purpose was to constrain the deformations of an object to plausible ones only. However, the final goal is different since, in Computer Vision, we seek to recover deformations, rather than simulate them. This bears similarities with the so-called inverse problem in the FEM, where one seeks to recover the forces that generated a certain shape, rather than to apply a force to deform an object.

Many variations of the physics-based models were proposed to reconstruct shapes from images. Among them, balloons [34, 35] were introduced to solve some issues of the original Snakes [84]. The key idea was to modify the external forces that deform the curve, and add an inflation force that makes the curve, or surface, expand. Deformable superquadrics [150, 108] were proposed to reconstruct more complex shapes by modeling both global and local deformations. Finally, another approach [105, 106] proposed to follow more closely the FEM formulation, and to model a deformable surface as a thin-plate under tension. Surveys of the different formulations are available in the field of medical imaging [107] and in a more general context [110]. Furthermore, the use of the Boundary Element Method has also been advocated to track deformable objects in 2D [54] and in 3D [55].

Modeling a surface as a finite element mesh yields representations that are of high dimensionality, since the meshes can be formed of many vertices, which, in conjunction with the computationally expensive FEM resolution methods, makes the problem impractical. To overcome these weaknesses, modal analysis emerged from the mechanical engineering community to reduce the number of degrees freedom by coupling the existing ones through vibration modes. These vibration modes are obtained by solving the generalized eigenproblem

$$\mathbf{K}\phi = \omega^2 \mathbf{M}\phi \ , \tag{2.2}$$

where $\mathbf{K}$ and $\mathbf{M}$ are the stiffness and mass matrices of Eq. 2.1, and the individual ϕ and ω define a mode and its frequency. The displacement of the mesh nodes is then given as

$$\mathbf{u} = \sum_{i=1}^{m} w_i \phi_i \ , \tag{2.3}$$

where w_i corresponds to the amplitude of mode i. In the full case, $m = 3\times$number of nodes. Since the modes with lower frequencies have more influence on the global shape of the surface, it is a valid approximation to discard the ones with higher frequencies, thus yielding a lower-dimensional problem.

Its ability to reduce the number of degrees of freedom quickly made modal analysis popular in the Computer Vision and Computer Graphics communities. Initially introduced for image segmentation [122, 123], it was also succesfully applied to medical imaging [114, 113, 115]. While computationally efficient, modal analysis as applied in computer vision assumes a constant stiffness matrix, which implies linearly elastic deformations. This however never is the case, since it is only true for barely visible deformations. Such models are therefore only rough approximations of the true non-linear behavior.

There has been some interest in better modeling the true physics of deformable objects via the non-linear finite element method in Computer Vision. However, unlike in Computer Graphics where one can tune the forces and material parameters that yield good deformations, recovering the shape of a surface from images requires a stable objective function to minimize. Some approaches have nonetheless been proposed for fitting a mesh to 3D range data [70, 80, 157], or for video-based shape recovery [158, 15]. However, these methods follow an analysis-by-synthesis approach, and manually set simulation parameters, until the ones that give the resulting shape that best matches the data are found. Only recently [75] has a true non-linear FEM formulation been proposed to recover the deformations of beam structures in the image plane, where image features act as forces, as was originally proposed for the Snakes. However, to the best of our knowledge no similarly accurate model has yet been demonstrated in the case of automatic 3D surface shape recovery from noisy image measurements.

2.1.4 Relations to our Work

The physics-based approach is very attractive, since it aims at modeling the true behavior of an object. Nonetheless, it suffers from several weaknesses. First, computing the appropriate matrices relies on the knowledge of material parameters which are often unknown. Second, building an accurate model is as much an art as a science, and even mechanical engineering experts often have to manually tune their models. Finally, optimizing such models yields computationally expensive algorithms whose objective functions exhibit many local minima.

However, modal analysis is strongly related to some of the approaches we studied in this work. Representing the deformations of a surface as a linear combination of modes yields effective low-dimensional models. Since this still requires potentially unknown physical constants, we will build such models following a statistical learning approach. Nonetheless, the general idea remains the same, and the resulting vibration modes look similar, as shown in Chapter 7. Furthermore, FEM models attempt to describe the local relationships between neighboring vertices of a mesh. Our local models follow a similar purpose, but in a statistical learning context.

2.2 Learned Global Models

Due to the complexity of accurately modeling the physics of highly deformable surfaces, statistical learning approaches have become an attractive alternative that takes advantage of observed training data. Active Shape Models (ASM) [38] are an early example of such models. They avoid the application-specific tayloring of the Snakes [84] to delinate shapes from images. Rather than guessing unknown material parameters, shape statistics are learned from available examples and used to constrain the deformations of contours.

2.2.1 Statistical Learning Methods

Many surface parameterizations rely on a large number of degrees of freedom. This, for example, is the case when specifying the shape of a triangulated surface in terms of its vertex coordinates. However, these degrees of freedom are often coupled and therefore lie on a much lower-dimensional manifold. Rather than explicitly adding constraints to the problem at hand, the core idea behind statistical learning is to express the problem in terms of its low dimensional representation, thus implicitly enforcing the constraints. The different methods are divided into linear and non-linear ones.

In the linear dimensionality reduction case, an example $\mathbf{y}$ is linked to its latent, possibly low-dimensional, representation $\mathbf{x}$ through the linear relationship

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{S}\mathbf{x} + \epsilon \; , \tag{2.4}$$

where $\mathbf{y}_0$ is the mean data value, and ϵ accounts for noise, usually taken as gaussian distributed. The matrix $\mathbf{S}$ contains the new basis vectors, which can be obtained by several different techniques.

The best-known method is Principal Component Analysis (PCA)[81], where the columns of $\mathbf{S}$ are taken as the eigenvectors of the data covariance matrix. In the context of non-rigid surfaces, this naturally sorts the deformations from low to high frequencies, as was the case with modal analysis. A probabilistic interpretation of PCA was also introduced [153], in which the distribution of the data in the new space is built from the eigenvalues of the data covariance matrix. Other standard examples of linear dimensionality reduction techniques are Independent Component Analysis (ICA) [36] where the basis is chosen as to minimize the dependencies between its components, and Canonical Correlation Analysis (CCA) [68] whose basis vectors are taken as the least correlated ones.

In many cases, however, the low-dimensional manifold onto which the training examples lie is not linear. Therefore, a linear model gives high probabilty to truly unlikely data, or vice-versa. As a result, several non-linear dimensionality reduction techniques were introduced. The initial method was kernel PCA [134], where PCA was applied in a higher dimensional feature space, related to the input space through a non-linear mapping based on kernel functions.

In a different context, several geometry-based techniques were proposed to retrieve the shape of the low-dimensional manifold. Isomap [149] and Locally Linear Embedding (LLE) [132] were introduced simultaneously, and both rely on k-nearest-neighbors to learn the latent representation. The former finds a low-dimensional space that enforces the geodesic distances between pairs of points to remain unchanged, whereas the latter assumes that a complex manifold is locally linear and tries to unfold it to a lower dimension. Other techniques such as Laplacian Eigenmaps [14] have since been proposed to overcome the weaknesses of Isomap or LLE. A common failure of these techniques is the absence of an inverse mapping from the low-dimensional space to the high-dimensional one. Such a mapping must then be learned separately, in terms of Radial Basis Functions (RBF) for example, which makes such non-linear techniques prone to errors both in the direct and the inverse mappings.

An alternative, possibly non-linear, learning technique is the Gaussian Process Latent Variable Model (GPLVM) [94], which was originally introduced as a generalization of probabilistic PCA. It introduces a mapping from the low-dimensional latent representation to the high-dimensional one written as,

$$\mathbf{y} = \sum_i w_i \phi_i(\mathbf{x}) + \epsilon \ , \tag{2.5}$$

where w_i are the weights of the possibly non-linear functions ϕ_i of the low-dimensional representation of the manifold $\mathbf{x}$. Since this mapping is linear in terms of w_i, these weights can be marginalized out. For any function k that results in a non-negative definite kernel matrix $\mathbf{K}$, such that $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, this marginalization yields a conditional density which corresponds to a product of D Gaussians with covariance $\mathbf{K}$, and can be written as

$$p(\mathbf{Y} \mid \mathbf{X}, \Theta) = \frac{1}{\sqrt{(2\pi)^{ND} |\mathbf{K}|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right) \ , \tag{2.6}$$

where $\mathbf{Y}$ and $\mathbf{X}$ are the matrices containing the N D-dimensional training examples and their latent representations respectively, and Θ contains the kernel hyper-parameters.

Several variations of the original GPLVM have been proposed. First, they have been adapted to account for dynamics, thus yielding the Gaussian Process Dynamical Model (GPDM) [169]. A weakness of the GPLVM is that the kernel function is defined between each of the latent variables of the training examples, which makes them computationally expensive. To overcome this issue, sparse representations have been proposed [96], where the kernel is defined in terms of a much smaller number of inducing variables. Finally, recently, a Hierarchical Gaussian Process Latent Variable Model (HGPLVM) was introduced [97] to express conditional independencies in the data.

2.2.2 Learned Models for Non-Rigid Modeling

In Computer Vision, the linear learning techniques quickly became very popular. The original Active Shape Models [38] were quickly extended to full 2D Active Appearance Models (AAM) [37, 104] to track 2D face deformations. In this case, the model is separated into shape and texture components, both modeled as linear combinations of basis vectors. Adaptations of this were also proposed to group appearance and shape in a single vector and to mix physics-based approaches with statistical learning [116]. The AAM were quickly turned into Morphable Models [19] designed to recover the full 3D shape of a face. They were used both to model the shape of the head of a new person in a neutral expression [131, 44] and to model various expressions of a same face [18]. They were combined to AAM to further account for appearance of the face instead of shape only [176].

The linear models have also been learned for non-rigid structure-from-motion [25]. The basic idea of this approach is to recover the 3D motion of points tracked throughout a video sequence. To prevent the 3D points from moving completely independently, their relative deformations are modeled as a linear combination of basis vectors. Such basis shapes can either be known a priori [1], or learned online [24, 156, 154] and their number automatically determined [10]. The standard approach relied on an orthographic camera model, but was extended to perspective camera [101, 164]. Other works showed how non-rigid structure-from-motion could also be used for recovering the relative motion of several rigid objects [177, 178]. A weakness of non-rigid structure-from-motion techniques is their sensitivity to missing data. Since they rely on tracked points, they typically tend to fail in real-life conditions, where points may disappear. Only very recently has this problem been alleviated by using hierarchical priors [155]. The remaining drawback of such methods is that they require a sufficiently long video sequence to infer deformation modes. This limits their applicability to relatively simple deformations.

In the context of articulated body tracking, statistical learning techniques have similarly been studied. A first approach [16] used linear models in 2D for tracking and recognizing hand gestures. This was further used to track the whole human body in specific motions, such as walking [140] or golf swings [159]. In these cases, PCA was applied to motion sequences rather than static poses. The human body models used were very rough approximations of true human shapes, which was improved upon since by using the SCAPE model [5, 7] that relies on PCA modes to describe body shape. Human body poses and motions were quickly found to be lying on highly non-linear manifolds. Therefore, several methods investigated the use of the GPLVM to model [57], and track [161, 117] such motions. Further applications were proposed to introduce dynamics in such non-linear models through the GPDM [160], and, recently, through the HGPLVM [4].

2.2.3 Relations to our Work

Learned models have proved very effective for many applications. They alleviate the need of unknown material parameters while yielding accurate representations of objects or materials statistics. However, some issues remain unsolved. First, gathering enough examples to build a meaningful database represents a very significant amount of work, especially in the case of highly deformable surfaces having many degrees of freedom. Second, registering the examples typically involves a painstaking process. In the case of faces [19] for example, laser scans first had to be aligned and then remeshed in order to have the same topology. Finally, all the proposed methods learn global models, which makes them valid only for a particular object. In the case of non-rigid surfaces, it would be more appropriate to model the behavior of a specific material than that of an individual surface, as is the case with physics-based approaches. These are some of the issues that we addressed in this work.

2.3 Alternative Shape Constraints

Physics-based models and statistical learning methods have been intensely researched for deformable surface modeling purposes. However, many different shape constraints and parameterizations have also been studied. Again, several of these approaches were first introduced in the Computer Graphics field for simulation purposes, and were later adapted to recover deformations from images.

2.3.1 Using Control Points

Modeling a deformable surface as a triangulated mesh typically yields many degrees of freedom. However, as mentioned earlier, many of these degrees of freedom are coupled, which can be enforced by using physics-based constraints or by representing the deformations as a combination of basis shapes. An alternative solution to modeling this coupling is to represent the motion of all mesh vertices as a function of a much smaller number of control points. The fine mesh is then obtained by interpolating the deformation between these control points.

One way to achieve this is through the use of Free-Form Deformations. Originally introduced for animation purposes [135], they were quickly adapted to recover shapes from images [43]. Interpolation can be done through Bézier volumes [39], polynomial curves [170], or B-splines [90, 48, 49]. A disadvantage of standard free-form deformations is their lack of ability to model local deformations. This was overcome by introducing Dirichlet Free-Form Deformations for animation of a hand [109], but also for computer vision purposes [72, 74]. A remaining drawback of free-form deformations techniques is that there is no automated way to create appropriate sets of control points.

An alternative to explicitly relying on control points that define the shape of a surface is the multi-resolution approach [66]. In this case, the deformation of an initial coarse mesh is computed, and, following a subdivision surface approach [30, 45], the mesh and its deformations are then refined. Several subdivision schemes have been proposed [102, 47, 87]. Such multi-resolution approaches were also used with dynamic vertex connectivity [88], and for mesh editing [184]. In the latter context, criticism were made that the editable regions were restricted by the initial coarse mesh. Laplacian surfaces [144, 181] were thus proposed to overcome this problem. To the best of our knowledge, multi-resolution methods have not been applied in the context of image-based shape recovery. A drawback of these techniques is that the surface is interpolated, which tends to yield visually pleasing results but may not correspond to what is observed in images.

2.3.2 Imposing Additional Constraints

Several constraints have been applied to the particular case of developable surfaces. The properties of such surfaces, such as isometry and vanishing Gaussian curvature have proved effective to recover their shape from single views [58]. More precise constraints, such as parallelism of lines, have been used to flaten curved documents in images [100]. A quasi-minimal parameterization of such surfaces was introduced [124]. In this work, the shape of a developable surface was modeled as a function of the end points of its directix lines. Finally, a recent work [125] proposed to set bounds on distances between feature points to reconstruct inextensible surfaces. However, this only reconstructed sparse points and the final surface shape was obtained with a thin-plate splines model, which relies on unknown parameters.

In this thesis, we will also study the use of inextensiblity constraints to help image-based recovery. As we will show later, representing the shape of a surface as a linear combination of modes does not enforce inextensibility. We will therefore prevent the mesh from shrinking or stretching by explicitly introducing such constraints into our algorithms. Additionally, we will introduce our own alternative constraints, and study how simple dynamics can help 3D reconstruction from video sequences. This approach differs from the existing techniques, and will result in two very generally applicable motion models.

2.4 Alternative Image Constraints

In addition to studying various shape constraints to improve the accuracy of 3D reconstruction, using as many sources of image information as possible has also been an important research direction. Most standard approaches, including ours, rely on texture, which will be presented in more details in the rest of this thesis. However, other image cues sometimes are available. The most popular ones are triangulation from multiple views, shading information, and silhouettes. They are usually combined and applied in conjunction with shape priors to make the reconstruction more robust.

In this work, we focused our research on the monocular case and relied on texture and silhouettes only. Working in such poor conditions will yield much more generally applicable models, which could be used in conjunction with additional sources of information. Ultimately, to build a working system, one should always rely on as much information as possible. However, by developing models that are effective in poor conditions, we ensure that they would still be effective in less challenging ones.

2.4.1 Multiple Views

Multiple view geometry has been of huge interest in Computer Vision to solve problems such as object pose estimation, relative camera pose estimation, and 3D shape reconstruction [60]. When such information is available, it therefore seems natural to rely on several views to reconstruct the shape of non-rigid surfaces, since it effectively constrains depth and mirrors what happens in human vision. Note, however, that deformation models are still useful to fill the untextured or occluded parts of the surface.

Stereo reconstruction relies on establishing correspondences between points across the different views. These methods have been applied to reconstruct abitrary scenes with no prior knowledge of their content [51, 29, 163]. Shape recovery has also been performed by fitting a known model to noisy stereo data [73]. A survey of multi-view reconstruction techniques can be found in [137].

Recently, there has been an increasing interest in relying on stereo to recover the complex shapes of clothes [146]. This constitues a very hard application, since the folds and wrinkles of clothing produce many self-occlusions, and make simple matching techniques fail. Various matching techniques have been proposed, such as spherical matching [145], as well as different shape representations such as Laplacian surfaces [42]. Such clothes motion capture has been succesfuly applied with specific markers printed on the garments [171], and very recently in the absence of such markers [23].

2.4.2 Shape from Shading

Another source of shape information is shading, especially for relatively untextured objects, such as faces that display few reliable feature points. Shape-from-shading [67] was originally formulated in the context of Lambertian surfaces of unknown albedo, with a single distant point light source. The goal of this approach was to recover 3D shape from a single image taken with a fully calibrated camera. Many variations [180] have been proposed since, but shape-from-shading algorithms still suffer from a number of limitations. Most of them depend on very restrictive assumptions that limit their applicability. Furthermore shape-from-shading is known to suffer from the Bas-Relief Ambiguity [13]. The same appearance can be obtained by applying a specific class of transformations to the shape and to the albedo of the surface. This is also true in the presence of several distant light sources or with an unknown viewpoint [179].

Various generalizations of the original Lambertian model were proposed [120, 129] to remove some of the most severe limitations. The use of interreflections, initially ignored, was introduced [118, 50]. They were shown to help solving for the Bas-Relief Ambiguity [31]. Additionally, the effects of shadows [89] and specularities [121] on the accuracy of the reconstruction were studied. Nowadays, the Lambertian assumptions tend to be replaced by more accurate models [3].

Shading information has been used in conjunction with other sources of information, such as stereo [17, 98]. This was successfully applied for cartographic modeling purposes in conjunction with a 3D mesh representation of the terrain surface [52]. Similarly, lighting information was combined with deformable models to reconstruct faces [133], and to recover the shape of non-rigid surfaces in 2D [173] as well as in 3D [172]. In this last work, shading was shown to help reducing the ambiguities arising from relying on texture only. Recently, shadows were used to improve human body pose recovery assuming sufficiently strong lighting such that they could act as a second image taken from a different camera [6].

Photometric stereo [174] also relies on lighting information, but differs from shape-from-shading by using several images taken under different lighting conditions. It is far more reliable and yields outstanding reconstructions when available, but requires an elaborate setup. As for shading, the influence of specularities were studied in photometric stereo [46]. Examples-based approaches were applied to recover the shape and material type of objects from images [63]. Recently, it was successfuly applied to the motion capture of clothes undergoing complex deformations through the use of colored lights [62].

2.4.3 Shape from Silhouettes

Silhouettes and contours of an object also provide excellent hints as to its shape. In the case of deformable surfaces, these silhouettes can either be true surface boundaries that physically exist or occluding contours that depend on the viewpoint. The standard approach is to detect these silhouettes in images and minimize the distance between them and the projected object contours. Extraction of the contours can be done by simple edge detection [28], or by more sophisticated methods such as Active Contours [84] or space carving [91].

In the context of non-rigid surfaces, occluding contours were used to reconstruct 3D objects from images [162, 32, 165, 22, 148]. They were used as external image forces to deform physics-based models [150]. They were also combined with texure for multiple view reconstruction [41]. In our own work [76, 77], we used an implicit surface formulation to detect occluding contours and use them to deform various objects such as an upper body, or a piece of paper in conjunction with inextensibility and smoothness constraints. Finally, silhouettes have also proved very effective for human body tracking [142, 2].

2.5 Summary

Many approaches have already been proposed to recover the shape of deformable surfaces from video. However, each of these approaches has its weaknesses. For physics-based models, material parameters must be known in advance, and accurate non-linear FEM models have proved too complex to be of practical use in our field. Existing learned models rely on training data, but without proposing convenient ways of generating them, and always are global models only applicable for a specific object. Finally, methods interpolating between control points provide no guarantee of modeling the true object's behavior, and constraints for developable surfaces are not generally applicable.

Nonetheless, the existing approaches also have strengths from which we can inspire our work. Modeling local relations between neighboring vertices, as in the physics-based approach, allows for re-usability of the models. Representing deformations as linear combinations of modes, as in modal analysis, yields effective low-dimensional models, which, when learned from data, do not even need material parameters. Finally, inextensibility constraints are very generally applicable, and, while not sufficient on their own, effectively help disambiguating the reconstruction problem.

In this thesis, we therefore propose to take advantage of the strengths of the existing methods, while overcoming their weaknesses. For this purpose, we will study several solutions to the reconstruction problem that range from very generic motion models to learned deformation models that more accurately correspond to a specific material. Furthermore, to remain generic, we will only rely on texture and edges. This will prevent us from having to make strong assumptions, as required by shape-from-shading, or to rely on multiple views which may not always be available. Nonetheless, when such information is at hand, nothing will prevent it to be introduced in our algorithms, thus yielding an even more robust system.

3 The Monocular 3D Reconstruction Problem

In this work, our overriding goal is to overcome the ambiguities inherent to monocular 3D deformable surface reconstruction and to incorporate solutions into robust algorithms. The first key step is therefore to formalize these ambiguities. To this end, we consider the case where the image information only comes from correspondences between a reference image for which we know the shape and the input image. This lets us write down the basic equations of shape recovery and exhibit its underlying ambiguities. Since, in practice, relying on correspondences only may not be sufficient, we present alternative image measurements that have been used in the experiments of this thesis. Nonetheless, these new measurements only avoid having additional ambiguities, and still leave the initial ones unsolved.

3.1 3D-to-2D Correspondences

The primary source of image information used in this work is texture. More specifically, most of our methods rely on correspondences between a reference image and an input image as depicted in Fig. 3.1. Several reasons motivated this choice. First, establishing correspondences between two images does not involve strong assumptions, apart from requiring the surface to be textured. This is in contrast with other approaches such as shape-from-shading. Second, given a reference image, it can be done from a monocular input and thus suits our purpose to remain as generic as possible. Finally, correspondences can be obtained from individual images, as opposed to a video sequence, which lets us go beyond developping tracking algorithms.

Detecting feature points in images has been of interest for a while in the Computer Vision community. In this work, we rely either on the SIFT keypoints detector [103] or on Harris's corners detector [59]. Once feature points have been detected in two images, they need to be matched to produce correspondences. When using SIFT, this can be done by a simple dot-product between specific vector representations of the feature points. For Harris's corners, methods based on randomized-trees have proved efficient [99]. From a large set of views obtained by applying random affine transformations to a reference image, a tree that models the relationships between neighboring keypoints is built. Each leaf-node of the tree then corresponds to a specific keypoint, and matching can done by dropping the feature points of a new image down the tree.

In both cases, we match the current image of interest with the reference image, in which the 3D shape and the camera calibration are known, as depicted in Fig. 3.2. Under such

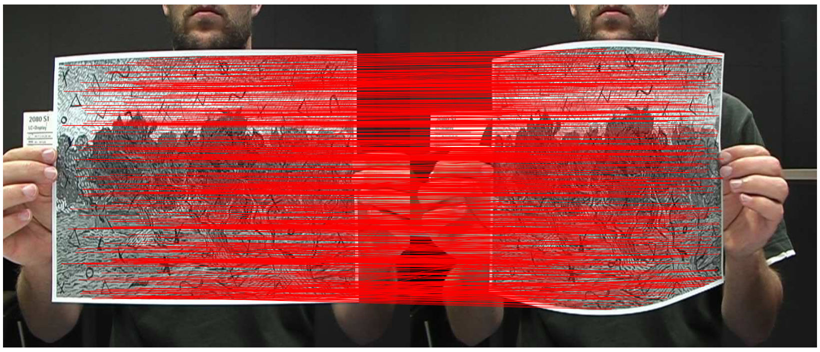


Figure 3.1: Correspondences between a reference image and an input image. The point-to-point correspondences are shown as red lines going from the first image to the second one.

assumptions, we can compute the 3D locations of the feature points on the reference image, by intersecting the ray between the camera center and the 2D image measurement with the facets of the triangulated mesh. This lets us represent a 3D point in terms of its barycentric coordinates with respect to the vertices of the facet intersected by the ray. This yields 3D-to-2D correspondences for the current image, where the 3D positions of the feature points are defined with respect to the unknown 3D positions of the mesh vertices. To recover the 3D shape, the idea is then to find the position of the mesh vertices that minimizes the distance between the detected 2D features and the 3D points locations projected into the image.

3.2 Single-Image Ambiguities

In this section, we show that recovering the 3D shape of a non-rigid surface from 3D-to-2D correspondences amounts to solving an ill-conditioned linear system. We then show that the degeneracies, or near-degeneracies, of this system correspond to depth ambiguities that can be explained in terms of a piecewise affine projection model. Since we use a single camera and assume its internal parameters to be known, we express all world coordinates in the camera referential for simplicity and without loss of generality.

3.2.1 Ambiguities under Perspective Projection

We now show how computing the 3D mesh vertex coordinates given 3D-to-2D correspondences can be formulated as the solution of a linear system and discuss its degeneracies. We start with a mesh containing a single triangle and extend our result to a complete one.

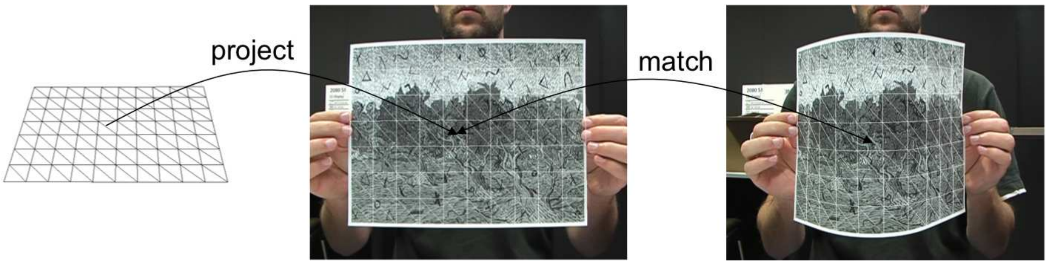


Figure 3.2: Obtaining image correspondences. A feature point is detected in the reference image, shown in the middle. Knowing the reference 3D shape of the mesh, on the left, and the camera projection matrix, we can retrieve the facet to which the feature point belongs, and define it in terms of its barycentric coordinates. The feature point can then be matched against points detected in the input image, shown on the right. This yields 3D-to-2D correspondences in terms of the unknown 3D mesh vertices in the input image.

3.2.1.1 Projection of a 3D Surface Point

Let $\mathbf{q}_i$ be a 3D point on the surface of interest. We write its perspective projection as

$$k_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{A} \left[\ \mathbf{R} \mid \mathbf{t} \right] \begin{bmatrix} \mathbf{q}_i \\ 1 \end{bmatrix} \ , \tag{3.1}$$

where $\mathbf{A}$ is the internal parameters matrix, and k_i a scalar accounting for depth. Since we assume that the camera and world referential are aligned, the camera rotation matrix $\mathbf{R}$ is the $3{\times}3$ identity matrix, $\mathbf{I}_{3\times 3}$, and the translation vector $\mathbf{t}$ is zero.

If $\mathbf{q}_i$ lies on the facet of a triangulated mesh, it can be expressed as a weighted sum of the facet vertices. Eq. 3.1 becomes

$$k_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{A}(a_i\mathbf{v}_1 + b_i\mathbf{v}_2 + c_i\mathbf{v}_3) \ , \tag{3.2}$$

where $\mathbf{v}_{i\ ,1\leq i\leq 3}$ are the vectors of 3D vertices coordinates and (a_i, b_i, c_i) the barycentric coordinates of $\mathbf{q}_i$.

3.2.1.2 Reconstructing a Single Facet

Let us assume that we are given a list of N_c^f such 3D-to-2D correspondences for points lying inside one single facet. The coordinates of its vertices $\mathbf{v}_{i\ ,1\leq i\leq 3}$ can be computed by solving the following equation where the k_i are treated as auxiliary variables to be recovered as well

$$\mathbf{M_f}\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ k_1 \\ \ldots \\ k_i \\ \ldots \\ k_{N_c^f} \end{bmatrix} = \mathbf{0}\ , \tag{3.3}$$

with

$$\mathbf{M_f} = \left[\begin{array}{ccc|ccccc} a_1\mathbf{A} & b_1\mathbf{A} & c_1\mathbf{A} & -\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} & 0 & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_i\mathbf{A} & b_i\mathbf{A} & c_i\mathbf{A} & 0 & \ldots & -\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{N_c^f}\mathbf{A} & b_{N_c^f}\mathbf{A} & c_{N_c^f}\mathbf{A} & 0 & \ldots & \ldots & \ldots & -\begin{bmatrix} u_{N_c^f} \\ v_{N_c^f} \\ 1 \end{bmatrix} \end{array}\right] .$$

For $N_c^f > 4$, if the columns of $\mathbf{M_f}$ had become linearly independent, the system would then have had a unique solution. However, this is not what happens.

To prove that $\mathbf{M_f}$ is rank-deficient, we show that its last column can always be written as a linear combination of the others as follows. From Eq. 3.2 we can write

$$-\begin{bmatrix} u_{N_c^f} \\ v_{N_c^f} \\ 1 \end{bmatrix} = a_{N_c^f}\mathbf{A}\lambda_1 + b_{N_c^f}\mathbf{A}\lambda_2 + c_{N_c^f}\mathbf{A}\lambda_3 \tag{3.4}$$

where $\lambda_j = -\mathbf{v}_j / k_{N_c^f}$ for $1 \leq j \leq 3$. For all $1 \leq i < N_c^f$, we have

$$\begin{aligned} a_i\mathbf{A}\lambda_1 + b_i\mathbf{A}\lambda_2 + c_i\mathbf{A}\lambda_3 &= -\frac{a_i}{k_{N_c^f}}\mathbf{A}\mathbf{v}_1 - \frac{b_i}{k_{N_c^f}}\mathbf{A}\mathbf{v}_2 - \frac{c_i}{k_{N_c^f}}\mathbf{A}\mathbf{v}_3 \\ &= -\frac{k_i}{k_{N_c^f}}\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} . \end{aligned}$$

This implies that the last column of the matrix $\mathbf{M_f}$ of Eq. 3.3 is indeed a linear combination of the previous ones with coefficients $(\lambda_1^T, \lambda_2^T, \lambda_3^T, -k_1/k_{N_c^f}, \ldots, -k_{N_c^f-1}/k_{N_c^f})$. In the general case, none of these coefficients is zero. Furthermore, because $\mathbf{A}$ has full rank and the barycentric coordinates are independent in general, the first 9 columns of $\mathbf{M_f}$ are linearly independent. Thus, given the particular structure of the right half of $\mathbf{M_f}$, trying to write any column as a linear combination of the others but the last one would yield wrong values on the last three rows, which could only be corrected by using the last column. This implies that, in general, $\mathbf{M_f}$ has full rank minus 1.

3.2.1.3 Reconstructing the Whole Mesh

If we now consider a mesh made of $N_v > 3$ vertices with a total of N_c correspondences well-spread over the whole mesh, Eq. 3.3 becomes

$$\mathbf{M_m} \begin{bmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_{N_v} \\ k_1 \\ \dots \\ k_{N_c} \end{bmatrix} = \mathbf{0} \,, \tag{3.5}$$

with

$$\mathbf{M_m} = \left[\begin{array}{cccccc|ccccc} a_1\mathbf{A} & b_1\mathbf{A} & c_1\mathbf{A} & 0 & \dots & \dots & -\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & b_j\mathbf{A} & c_j\mathbf{A} & d_j\mathbf{A} & 0 & \dots & 0 & -\begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_l\mathbf{A} & 0 & c_l\mathbf{A} & 0 & e_l\mathbf{A} & \dots & 0 & \dots & -\begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array} \right] .$$

Coefficients similar to those of Eq. 3.4 can be derived to compute $[u_{N_c}, v_{N_c}, 1]^T$ as a linear combination of the non-zero columns of the last row. Since these coefficients only depend on k_{N_c}, on the mesh vertices and on the projection matrix, it can easily be checked that, as in the single triangle case, the last column of the matrix can be expressed as a linear combination of the others, which then are linearly independent. Thus matrix $\mathbf{M_m}$ of Eq. 3.5 has still full rank minus 1. This reflects the well-known scale ambiguity in monocular vision.

Representing the problem as in Eq. 3.5 was convenient to discuss the rank of the matrix. However, in practice, we want to recover the vertex coordinates but are not interested in having the k_i as unknowns. We therefore eliminate them by rewriting Eq. 3.5 as

$$\mathbf{M} \begin{bmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_{N_v} \end{bmatrix} = \mathbf{0} \,, \tag{3.6}$$

with

$$\mathbf{M} = \begin{bmatrix} a_1\mathbf{T_1} & b_1\mathbf{T_1} & c_1\mathbf{T_1} & 0 & ... & ... \\ ... & ... & ... & ... & ... & ... \\ 0 & b_j\mathbf{T_j} & c_j\mathbf{T_j} & d_j\mathbf{T_j} & 0 & ... \\ ... & ... & ... & ... & ... & ... \\ a_l\mathbf{T_l} & 0 & c_l\mathbf{T_l} & 0 & e_l\mathbf{T_l} & ... \\ ... & ... & ... & ... & ... & ... \end{bmatrix} \text{ , and } \mathbf{T_i} = \mathbf{A}_{2\times 3} - \begin{bmatrix} u_i\mathbf{A}_3 \\ v_i\mathbf{A}_3 \end{bmatrix} \text{ ,}$$

where $\mathbf{A}_3$ represents the last row of matrix $\mathbf{A}$ and $\mathbf{A}_{2\times 3}$ its first two rows. By construction, $\mathbf{M}$ has the same rank as matrix $\mathbf{M_m}$, therefore the following results are valid for both representations of the problem.

3.2.1.4 Effective Rank

In the previous paragraph, we showed that $\mathbf{M_m}$ has at most full rank minus one. However, this does not tell the whole story: In general, it is ill-conditioned and many of its singular values are small enough so that, in practice, it should be treated as a matrix of even lower rank. To illustrate this point, we projected randomly sampled points on the facets of the synthetic 88-vertices mesh of Fig. 3.3 (a) using a known camera model. We then computed the singular values of $\mathbf{M}$, which we plot in Fig. 3.3 (b).

In Fig. 3.4, we show the effect of adding two of the corresponding singular vectors—one associated to the zero singular value and the other to a small one—to the mesh in its reference position.

Even though only one of these values is exactly zero, we can see that they drop down drastically after the first $2N_v = 176$. This shows that, even though the matrix may have full rank minus 1, the solution of the linear system would be very sensitive to noise. Therefore, in a real situation, we would actually be closer to having N_v ambiguities, which can be understood in terms of the piecewise affine model we introduce below.

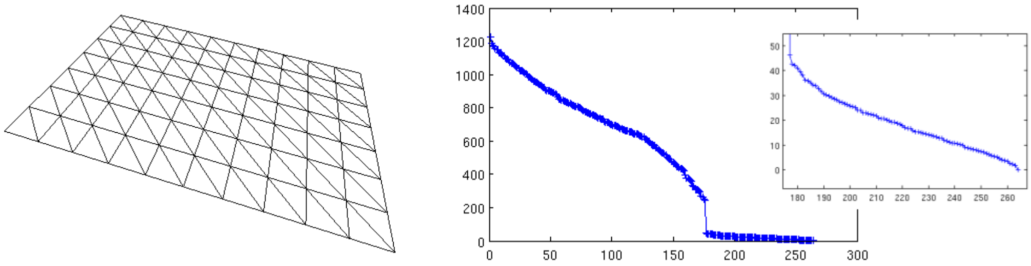
3.2.2 Ambiguities under Piecewise Affine Projection

A piecewise affine camera model is one that involves an affine transform for each facet of the mesh. This approximation is warranted if the facets are small enough to neglect depth variations across them.

3.2.2.1 Projection of a 3D Surface Point

As in the perspective case, let $\mathbf{q}_i$ be a 3D point whose coordinates are again expressed in the camera referential. We write its projection to a 2D image plane as

$$k \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \mathbf{P}'\mathbf{q}_i \ , \quad \mathbf{P}' = \mathbf{A}' \left[\ \mathbf{I}_{2\times 2} \ \middle| \ \mathbf{0} \ \right] \qquad (3.7)$$



(a) (b)

Figure 3.3: Effective rank of matrix $\mathbf{M}$. (a) 88-vertices mesh seen from the same viewpoint as the one used for reconstruction. (b) Singular values of $\mathbf{M}$ for the mesh of (a). Note how the values drop down after the $2N_v = 176^{th}$ one, as predicted by the affine model of Section 3.2.2. The small graph on the right is a magnified version of the part of the graph containing the small singular values. The last one is zero up to the precision of the matlab routine used to compute it and the others are not very much larger.

where k is a depth factor associated to the affine camera and $\mathbf{A}'$ is a 2×2 matrix representing the internal parameters. As in Section 3.2.1, we study the ambiguities for a mesh containing first a single triangle and then many.

3.2.2.2 Reconstructing a Single Facet

We can again write a linear system for a single triangle containing N_c^f 3D-to-2D correspondences, with 3D points given by their barycentric coordinates

$$\left[\begin{array}{ccc|c} a_1\mathbf{P}' & b_1\mathbf{P}' & c_1\mathbf{P}' & -\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} \\ \cdots & \cdots & \cdots & \cdots \\ a_i\mathbf{P}' & b_i\mathbf{P}' & c_i\mathbf{P}' & -\begin{bmatrix} u_i \\ v_i \end{bmatrix} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N_c^f}\mathbf{P}' & b_{N_c^f}\mathbf{P}' & c_{N_c^f}\mathbf{P}' & -\begin{bmatrix} u_{N_c^f} \\ v_{N_c^f} \end{bmatrix} \end{array}\right] \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ k \end{bmatrix} = \mathbf{0} \; . \qquad (3.8)$$

Since we only have one facet, we also only have one projection matrix, therefore a single k corresponding to the average depth of the facet is necessary, and all $[u_i, v_i]^T$ can be put in the same column.

Since $\mathbf{P}'$ is of size 2×3, it has at most rank 2. Moreover, we can show that the last

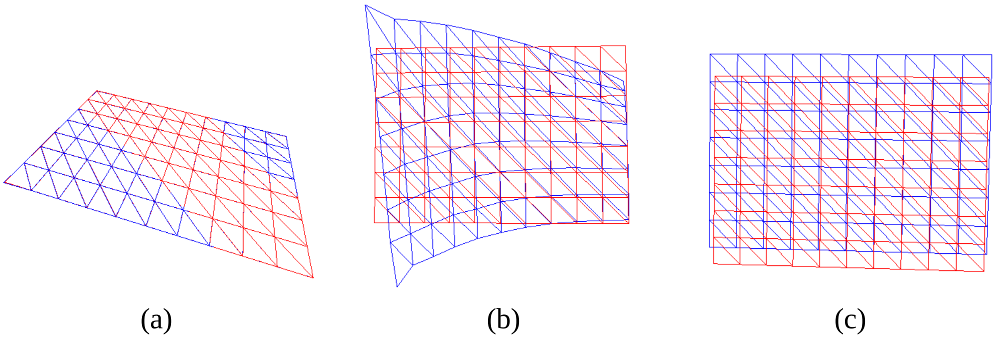


Figure 3.4: Visualizing vectors associated to small singular values. (a) Reference mesh and mesh to which one the vectors has been added seen from the original viewpoint, in which they are almost indistinguishable. (b) The same two meshes seen from a different viewpoint. (c) The reference mesh modified by adding the vector associated to the zero singular value. Note that the resulting deformation corresponds to a global scaling.

column of the global matrix also is a linear combination of the two first columns of $\mathbf{P}'$

$$\begin{aligned}\begin{bmatrix} u_i \\ v_i \end{bmatrix} &= \mathbf{P}'\frac{1}{k}(a_i\mathbf{v}_1 + b_i\mathbf{v}_2 + c_i\mathbf{v}_3) \\ &= \left[\ \mathbf{A}' \mid \mathbf{0}\ \right]\frac{1}{k}(a_i\mathbf{v}_1 + b_i\mathbf{v}_2 + c_i\mathbf{v}_3) \\ &= \frac{a_i}{k}\mathbf{A}'\begin{bmatrix} \mathbf{v}_{1,1} \\ \mathbf{v}_{1,2} \end{bmatrix} + \frac{b_i}{k}\mathbf{A}'\begin{bmatrix} \mathbf{v}_{2,1} \\ \mathbf{v}_{2,2} \end{bmatrix} + \frac{c_i}{k}\mathbf{A}'\begin{bmatrix} \mathbf{v}_{3,1} \\ \mathbf{v}_{3,2} \end{bmatrix}. \end{aligned} \tag{3.9}$$

The coefficients of Eq. 3.9 are independent of the correspondence considered and are therefore valid for any row i of the matrix. This finally means that, when $N_c^f \geq 3$, the rank of the matrix of Eq. 3.8 is always 6.

3.2.2.3 Reconstructing the Whole Mesh

As discussed above, when there are several triangles, using the piecewise affine model amounts to introducing a projection matrix per facet. However, since in reality we only have one camera, its internal parameters, rotation matrix, and center are bound to be the same for each triangle. This only lets us with a variable depth factor k_i for each facet i among the N_f facets of the mesh. We can then write the system

$$\mathbf{M}'_\mathbf{m}\begin{bmatrix} \mathbf{v}_1 \\ \cdots \\ \mathbf{v}_{N_v} \\ k_1 \\ \cdots \\ k_{N_f} \end{bmatrix} = \mathbf{0}, \tag{3.10}$$

with

$$\mathbf{M'_m} = \left[\begin{array}{cccccc|ccccc} a_1\mathbf{P}' & b_1\mathbf{P}' & c_1\mathbf{P}' & 0 & ... & ... & -\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} & 0 & ... & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & b_j\mathbf{P}' & c_j\mathbf{P}' & d_j\mathbf{P}' & 0 & ... & 0 & -\begin{bmatrix} u_j \\ v_j \end{bmatrix} & 0 & ... & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \\ a_l\mathbf{P}' & 0 & c_l\mathbf{P}' & 0 & e_l\mathbf{P}' & ... & 0 & ... & -\begin{bmatrix} u_l \\ v_l \end{bmatrix} & 0 & ... \\ ... & ... & ... & ... & ... & ... & ... & ... & ... & ... & ... \end{array} \right] .$$

The left half of $\mathbf{M'_m}$, which is of size $2N_c \times 3N_v$, N_c being the total number of correspondences, has at most rank $2N_v$ because $\mathbf{P}'$ has rank 2. Similarly, its right half, which is of size $2N_c \times N_f$, has at most rank $N_f - 1$, because we can again show that its last column is a linear combination of the previous ones in a similar manner as was done for the perspective case, with the coefficients of Eq. 3.9. This means that for a full mesh, $\mathbf{M'_m}$ has at most rank $2N_v + N_f - 1$. This leaves us with $N_v + 1$ ambiguities. This again seems natural due first to the same scale ambiguity as in the perspective case, and second to the fact that now each vertex is free to move along the line of sight. This number corresponds to the number observed in the perspective case of Section 3.2.1.4, except that, in the affine case, a global scale is different from all vertices sliding along the line of sight, which produces an extra zero singular value.

3.3 Image Information in Practice

In the previous section, we formalized the ambiguities of monocular shape recovery using 3D-to-2D correspondences. We showed that, even though under perspective projection we only have a single true ambiguity, we are close to having the same depth ambiguities as with a piecewise affine camera model, that is one ambiguity per mesh vertex. Furthermore, these ambiguities appear when we have dense correspondences over the whole mesh. However, this rarely is the case in practice. Here, we present practical ways of exploiting the information available from the image. Throughout this work, we will use two complementary sources of image information: Texture and silhouettes. One constrains the interior of the surface while the others give us information about its boundaries and its occluding contours. In particular, using border information helps constraining the boundary vertices of the mesh, which belong to fewer facets than those in the middle and are therefore less constrained by texture.

3.3.1 Texture

There are two different ways of exploiting texture information. Using correspondences detected as interest points, as explained at the beginning of this chapter, or through template matching that uses the intensity of the whole image. We first describe the latter, and then show how it can also serve to obtain dense correspondences over the whole surface.

3.3.1.1 Template Matching

To exploit texture information as comprehensively as possible, one may also rely on image intensity rather than interest points. This can be done through template matching by treating each mesh facet as an independent template, sampling its barycentric coordinates, and observing the underlying image pixels intensities. The values of the samples in the current image therefore depend on the 3D positions of the mesh vertices. Shape recovery is then done by maximizing the normalized cross-correlation between the templates in the reference image and in the current image, which can be written as

$$E_{TM}(\mathbf{y}) = \sum_{j=1}^{N_f} \gamma(P(\mathbf{y}_{\text{ref}}, j, \mathbf{I}_{\text{ref}}), P(\mathbf{y}, j, \mathbf{I})) \ , \tag{3.11}$$

where N_f is the number of facets, and $P(\mathbf{y}, j, \mathbf{I})$ is the projection of the j^{th}facet of surface $\mathbf{y}$ in image $\mathbf{I}$. γ denotes the normalized cross-correlation function, which is expressed as

$$\gamma(T, F) = \frac{\sum_{i \in Samples}(F(x_i, y_i) - \bar{F})(T(u_i, v_i) - \bar{T})}{(\sum_{i \in Samples}(F(x_i, y_i) - \bar{F})^2 \sum_{i \in Samples}(T(u_i, v_i) - \bar{T})^2)^{1/2}} \ , \tag{3.12}$$

where T and F are the reference template and the projected input facet for which we sample the barycentric coordinates, and $\bar{T}$ and $\bar{F}$ indicate their mean intensity values.

It may seem that template matching would better constrain shape recovery than correspondences, and therefore would give fewer ambiguities. However, it only is equivalent to having extremely dense correspondences, that is one for each pixel covered by the surface. This is very close to the case we studied in the previous section, where we densely sampled the barycentric coordinates of a synthetic mesh. This implies that depths would still be ill-constrained, which leaves us with the same number of ambiguities

The advantage of template matching is that it can be used even with poorly textured surfaces. However, it typically suffers from local minima, since different locations in the image can look alike. Therefore it can only be applied for tracking, where the initial guess is close to the true solution. This is not the case when using correspondences, since a distance function is much easier to optimize.

3.3.1.2 Obtaining Dense Correspondences

To reconstruct surfaces from individual images, template matching is not practical and correspondences should therefore be favored. Nonetheless, as mentioned above, the surface deformations are often such that only few matches between the reference configuration and the current image can be established using the techniques presented at the beginning of this chapter. Since these might not be sufficient to fully recover the 3D shape, we take advantage of the fact that non-rigid 2D registration is much better constrained than full 3D shape recovery to obtain dense correspondences. Given the 2D deformations of the surface in the image plane, we sample the barycentric coordinates of the facets of the 3D reference mesh. Starting from the same barycentric coordinates, but this time in the 2D results, we look for the square image patch most similar to the corresponding one in the reference image by computing their normalized cross-correlation. Assuming a correct 2D registration, we obtain matches over the whole surface, that can be slightly misplaced, but not completely wrong. We now explain how such a 2D registration can be obtained.

3.3.1.3 2D Registration

2D registration has also been of interest in the Computer Vision community. Different types of shape constraints have been proposed for this purpose, such as deformations based on Radial Basis Functions [11], or reasoning on self-occlusions [53]. Simple smoothness constraints based on the neighborhood of mesh vertices have also been used for non-rigid surface detection [127, 182]. In the presence of a video sequence, the most effective way of obtaining an accurate 2D registration is by tracking the surface in 2D using template matching in a similar manner as for the 3D case. Nonetheless, we will show that this can be done in closed-form when no video sequence is available.

Most ideas that were introduced in the initial linear formulation of 3D reconstruction are still valid for the 2D case. The surface is still represented as a triangulated mesh, though now each of its vertices only has 2 coordinates. We assume that we are given an initial set of sparse correspondences that relate points on the mesh to image locations, which can be obtained as described in Section 3.1. Similarly as in the 3D case, a point on the mesh can be defined by its barycentric coordinates with respect to the facet it belongs to.

Within this framework, it can easily be checked that we can again write the registration problem as the solution of a linear system of the form

$$\mathbf{M_p}\mathbf{u} = \mathbf{0} \; , \tag{3.13}$$

where $\mathbf{M_p}$ is the $2N_c \times 2N_v$ matrix equivalent to $\mathbf{M}$ in Eq. 3.6, and $\mathbf{u}$ is the vector of concatenated 2D mesh coordinates (x_i, y_i).

Although in Section 3.2 we claimed that only depths were ambiguous, and therefore 2D registration could be done by directly solving this system, this is only true with feature points covering the whole surface. In many real applications, only few matches can be obtained, and therefore the system becomes ill-conditioned. To overcome this issue, we propose to introduce a similar regularization as in [127]. Assuming that the mesh topology is regular, as is the case of hexagonal meshes, we can penalize the bending of a triplet of neighboring vertices $(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$ that form a line in the rest configuration by minimizing the deformation energy

$$E_i = (2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 \; . \tag{3.14}$$

Since we solve a linear system in the least-squares sense, these equations can be integrated into our framework by separating their x- and y-contributions. We can therefore write our problem as

$$\begin{bmatrix} \mathbf{M_p} \\ w_b\mathbf{B} \end{bmatrix} \mathbf{u} = \mathbf{0} \; , \tag{3.15}$$

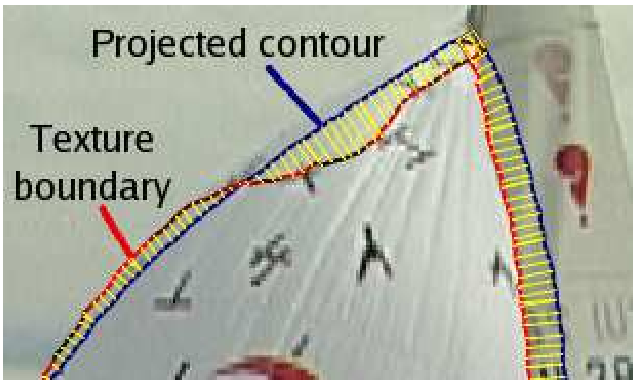


Figure 3.5: Using silhouettes information. Texture edges, shown in red, are detected in the current image. The contour of the current deformed mesh is projected onto the image, here depicted in blue. We then look in the direction of the normal of this projected contour for a detected image edge. Finally, we minimize the distance between the image and projected contours. Note that edge detection yields wrong edges. Our algorithms must therefore be robust to such erroneous information.

where $\mathbf{B}$ is the $2N_t \times 2N_v$ matrix encoding the regularization equations for the N_t triplets of the mesh, and whose coefficients have values 2, or -1. w_b is the weight that sets the relative influence of the correspondence equations and the regularization equations. In practice, solving this system proved sufficient to obtain a good 2D registration that allowed us to compute the dense 3D-to-2D correspondences required for 3D reconstruction.

One could think that such a regularization could also be used for 3D reconstruction. However, whereas it is true that deformations in x- and y- directions should be equally smooth, this is not the case for the z-direction where we expect to see larger deformations. One could then try to overcome this issue by less penalizing the bending in this direction. However, this would yield constraints that are sensitive to the global rotation of the surface.

3.3.2 Silhouettes

In several cases, we use silhouette information in conjunction with texture. This is crucial near the boundary of the mesh. The vertices in the center of the mesh are relatively well constrained by texture, since the texture of their six neighboring facets provides information about their position. By contrast, the boundary vertices are only influenced by at most three facets, and can therefore often move more freely. Silhouettes can also help in the presence of occluding contours, when a part of the texture is not visible anymore. Such contours provide information about the surface normals. Detecting an occluding contour can be done by sophisticated implicit surface representations [76]. In this work, we use a much simpler approach based on OpenGL rendering of the facets. Using a z-buffer technique, we can find which facets are hidden by other ones, and thus detect the occuding contours.

Boundaries of the mesh and occluding contours are then used as depicted by Fig. 3.5. We detect edges in the current image by applying a simple Canny edge detector [28], or a more sophisticated technique [138]. We then project the 3D silhouettes of our mesh in the image, and sample them. For each such sample, we look in the direction of its 2D normal for detected image edge points. The shape is finally recovered by minimizing the distance between the sample and its, possibly multiple, candidate points. This assumes that the true edge will be the strongest one in the image, and therefore will pull the surface away from wrong candidates.

Whereas texture and silhouettes are complementary, the latter require having a good initial guess, since we detect edges around the current position. Trying to match edges detected in the whole image would lead to local minima, and is therefore not practical. For this reason, we only used silhouette information for tracking purposes. However, they could be included in our other approaches to refine the solutions found from texture only.

3.4 Conclusion

In this chapter, we have shown that recovering the 3D positions of the vertices of a triangulated mesh from a single viewpoint using dense correspondences amounts to solving an ill-constrained linear system. More specifically, we have shown that the ambiguities of the problem correspond to estimating the depths of the mesh vertices. Furthermore, we have presented practical ways of exploiting image information. However, even for an ideally textured surface, correspondences and silhouettes will never fully constrain the reconstruction. This leaves us with an ill-posed problem, that requires some knowledge *a priori* of the plausible deformations of the surface of interest. In the coming chapters, we will propose several ways of introducing such missing knowledge.

4 Motion Models

In this chapter, we introduce a first type of constraints to improve the conditioning of shape recovery from a single viewpoint. We propose two different motion models that will link the deformations of a surface from one frame to the next. The first is a simple linear motion model that directly addresses the ambiguities observed in the previous chapter, whereas the second gives a more plausible approximation to the real dynamical behavior of an object, and lets us formulate shape recovery as a convex optimization problem.

Whereas deformation models that penalize unlikely shapes are only valid, in the best case, for a particular material, motion models are much more generally applicable. They follow the idea that the motion of the mesh vertices from one frame to the next cannot be random, but depends on the previous shape. Such motion can therefore be described by a particular dynamical model, that approximates the true behavior more or less accurately. These models predict the location of vertices in the current frame based on the results in the previous ones. The actual shape is then taken to be the one that best matches such a prediction, while also minimizing the reprojection error.

4.1 Linear Motion Model

A very simple approach is to introduce a minimal set of constraints specifically designed to overcome the ambiguities discussed in the previous chapter. Since the linear systems of Section 3.2 are rank-deficient, we need to introduce additional constraints to obtain acceptable solutions. We show here that frame-to-frame motion can be expressed as a set of additional linear constraints that make our linear systems well-conditioned, first in the affine case and then in the projective one. We therefore perform the reconstruction over several frames simultaneously and simply limit the range of motion from one frame to the next.

4.1.1 Constraining the Affine Reconstruction

Given a temporal sequence of N_I images and the corresponding matrices ${\mathbf{M}'_\mathbf{m}}^t$, $1 \leq t \leq N_I$ of Eq. 3.10, we can create a block diagonal matrix whose blocks are the ${\mathbf{M}'_\mathbf{m}}^t$ and use it to write a big linear system that the vertex coordinates in all frames must satisfy simultaneously. However, without temporal consistency constraints, the ambiguities remain: As discussed in Section 3.2.2, when the camera coordinates are aligned with the world coordinates, reconstruction is only possible up to an unknown motion along the z-axis for each vertex at each time step. To mitigate this problem, it is therefore natural to link the z value

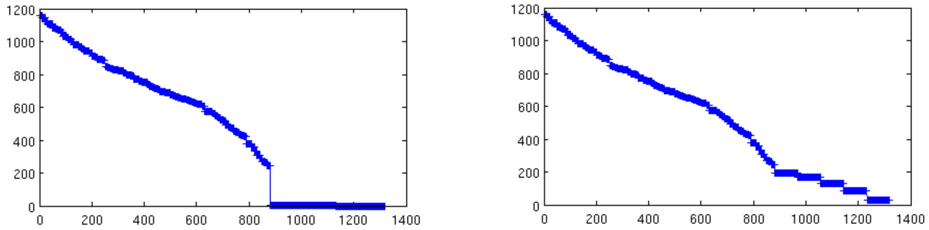


Figure 4.1: Singular values for a 5 frames sequence under affine projection. Left: Without temporal consistency constraints between frames, the linear system has many zero singular values, which implies severe reconstruction ambiguities. Right: Constraining the z coordinates as discussed in Section 4.1.1 leaves the non zero singular values unchanged but increases the value of the others, thus removing the ambiguities.

of vertices across time. The simplest way to do this is to write

$$\mathbf{v}_z^{t+1} - \mathbf{v}_z^{t} = 0 \tag{4.1}$$

for all vertices and all times. These constraints and those imposed by the 3D-to-2D correspondences can then be imposed simultaneously by solving with respect to Θ'

$$\mathbf{M}'_{\mathbf{s}}\Theta' = \mathbf{b}' \ , \tag{4.2}$$

where

$$\begin{aligned}
\Theta' &= \begin{bmatrix} \mathbf{v}_1^{1\,T} & \dots & {\mathbf{v}_{N_v}^{1}}^T & k_1^1 & \dots & k_{N_f}^1 & \dots & {\mathbf{v}_1^{N_I}}^T & \dots & {\mathbf{v}_{N_v}^{N_I}}^T & k_1^{N_I} & \dots & k_{N_f}^{N_I} \end{bmatrix}^T , \\
\mathbf{b}' &= \begin{bmatrix} \mathbf{0} & z_1^{first} & \dots & z_{N_v}^{first} & \mathbf{0} \end{bmatrix}^T , \\
\mathbf{M}'_{\mathbf{s}} &= \begin{bmatrix}
{\mathbf{M}'_{\mathbf{m}}}^1 & \mathbf{0} & \dots & \dots & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\dots & \mathbf{0} & {\mathbf{M}'_{\mathbf{m}}}^t & \mathbf{0} & \dots & \dots \\
\dots & \dots & \mathbf{0} & {\mathbf{M}'_{\mathbf{m}}}^{t+1} & \mathbf{0} & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\dots & \dots & \dots & \dots & \mathbf{0} & {\mathbf{M}'_{\mathbf{m}}}^{N_I} \\
\mathbf{C} & \mathbf{0} & \dots & \dots & \dots & \dots \\
-\mathbf{C} & \mathbf{C} & \mathbf{0} & \dots & \dots & \dots \\
\mathbf{0} & -\mathbf{C} & \mathbf{C} & \mathbf{0} & \dots & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\mathbf{0} & \dots & \dots & \mathbf{0} & -\mathbf{C} & \mathbf{C}
\end{bmatrix} ,
\end{aligned}$$

z_i^{first} is the z-coordinate of vertex i in the first frame, in which we assume that the shape is known, and $\mathbf{C}$ is an $N_v \times 3N_v$ matrix containing a single 1 in each row, which corresponds to the z-coordinate of one vertex.

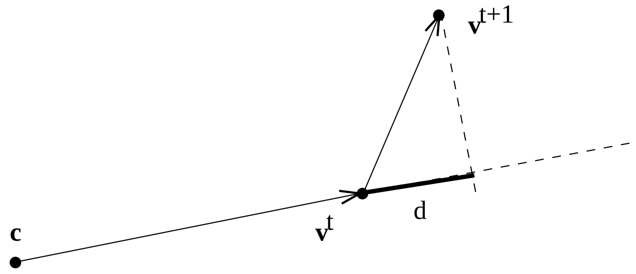


Figure 4.2: Under a perspective camera model, the depth d of a vertex $\mathbf{v}$ at time $t+1$ can be obtained by projecting vector $\mathbf{v}^t\mathbf{v}^{t+1}$ on the line-of-sight. This is done through a dot-product between this vector and the normalized direction of the line-of-sight, which is a non-linear function of the vertex position.

The number of constraints we add in this manner is equal to the number $N_v \times N_I$ of ambiguities that we derived in Section 3.2.2. Therefore it affects the rank of $\mathbf{M}'_{\mathbf{s}}$, and reduces the number of ambiguities to zero as shown in Fig. 4.1. Moreover, these constraints do not overlap with the ones imposed by the correspondences and can then be considered as minimal.

4.1.2 Constraining the Perspective Reconstruction

In Section 3.2.2, we showed that ambiguities under perspective projection are similar to those under piecewise affine projection. It is therefore natural to constrain the reconstruction in a similar way, that is by limiting the motion along the line-of-sight. However, since it is not parallel to the z-axis anymore, the constraints become more difficult to express.

Let us consider one vertex $\mathbf{v}$ of the mesh at times t and $t+1$. We can try minimizing,

$$d = \mathbf{v}^t\mathbf{v}^{t+1} \cdot \mathbf{e}^t \ , \ \text{with } \mathbf{e}^t = \frac{\mathbf{c}\mathbf{v}^t}{\|\mathbf{c}\mathbf{v}^t\|} \tag{4.3}$$

the length of the projection on the line-of-sight of the vector $\mathbf{v}^t\mathbf{v}^{t+1}$, where $\mathbf{c}$ represents the optical center of the camera, as depicted by Fig. 4.2. The difficulty comes from the fact that this constraint is non-linear and can therefore not be introduced into our linear formulation. We overcome this problem by replacing the exact formulation of d by an upper bound that can be expressed linearly as follows:

$$\begin{aligned}
d^2 &= (\mathbf{v}^t\mathbf{v}^{t+1} \cdot \mathbf{e}^t)^2 \ , \\
&= (\mathbf{e}^t_x(x_c^{t+1} - x_c^t) + \mathbf{e}^t_y(y_c^{t+1} - y_c^t) + \mathbf{e}^t_z(z_c^{t+1} - z_c^t))^2 \ , \\
&\leq (\mathbf{e}^t_x(x_c^{t+1} - x_c^t))^2 + (\mathbf{e}^t_y(y_c^{t+1} - y_c^t))^2 + (\mathbf{e}^t_z(z_c^{t+1} - z_c^t))^2 \\
&\quad + (\mathbf{e}^t_x(x_c^{t+1} - x_c^t))^2 + (\mathbf{e}^t_y(y_c^{t+1} - y_c^t))^2 \\
&\quad + (\mathbf{e}^t_x(x_c^{t+1} - x_c^t))^2 + (\mathbf{e}^t_z(z_c^{t+1} - z_c^t))^2 \\
&\quad + (\mathbf{e}^t_y(y_c^{t+1} - y_c^t))^2 + (\mathbf{e}^t_z(z_c^{t+1} - z_c^t))^2 \ , \\
&\leq (3\sin(\theta_x^{max})(x_c^{t+1} - x_c^t))^2 \\
&\quad + (3\sin(\theta_y^{max})(y_c^{t+1} - y_c^t))^2 \\
&\quad + (3(z_c^{t+1} - z_c^t))^2 \ , \qquad (4.4)
\end{aligned}$$

where x_c, y_c and z_c are the coordinates of a vertex in the camera reference system, and θ_x^{max} and θ_y^{max} are the maximum angles between the camera center and the points projecting on the left/right, and upper/lower border of the image, respectively.

As in Section 4.1.1, these constraints and those imposed by the 3D-to-2D correspondences can be imposed simultaneously. We rewrite Eq. 4.2 as

$$\mathbf{M_s}\Theta = \mathbf{b} \ , \qquad (4.5)$$

where

$$\begin{aligned}
\Theta &= \left[\begin{array}{ccccccc} \mathbf{v}_1^{1T} & ... & \mathbf{v}_{N_v}^{1\ T} & ... & \mathbf{v}_1^{N_I T} & ... & \mathbf{v}_{N_v}^{N_I T} \end{array} \right]^T \ , \\
\mathbf{b} &= \left[\begin{array}{ccccc} \mathbf{0} & \mathbf{v}_1^{first T} & ... & \mathbf{v}_{N_v}^{first T} & \mathbf{0} \end{array} \right]^T \ ,
\end{aligned}$$

and $\mathbf{M_s}$ is built by replacing in $\mathbf{M'_s}$ the matrices $\mathbf{M'_m}^t$ of Eq. 3.10 by the matrix $\mathbf{M}$ of Eq. 3.6 and the $\mathbf{C}$ matrices by $3N_v \times 3N_v$ matrices, containing a single value in each row that will constrain the x-, y-, or z-coordinate of one vertex. This value is set to one of the three coefficients of Eq. 4.4, depending on which coordinate the row corresponds to.

Fig. 4.3 shows how the singular values of the system are affected by introducing our depth constraints. As in the affine case, we can see that the smaller singular values have increased and are now clearly different from zero. Since this was our only goal in adding constraints, this justifies our approach to liberalization by minimizing the upper bound of d of Eq. 4.4 instead of d itself. Note that because we added more equations than was strictly necessary, the other singular values also increased, but only very slightly.

In practice the correspondences are never perfect and include noise and outliers. We therefore solve Eq. 4.5 in the least-squares sense and take Θ to be

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \ (\mathbf{M_s}\Theta - \mathbf{b})^{\mathrm{T}}\mathbf{W}(\mathbf{M_s}\Theta - \mathbf{b}) \ , \qquad (4.6)$$

where $\mathbf{W}$ is a diagonal matrix of ones for the lines corresponding to projection constraints and a user-defined weight for those that correspond to the depth constraints. The weight

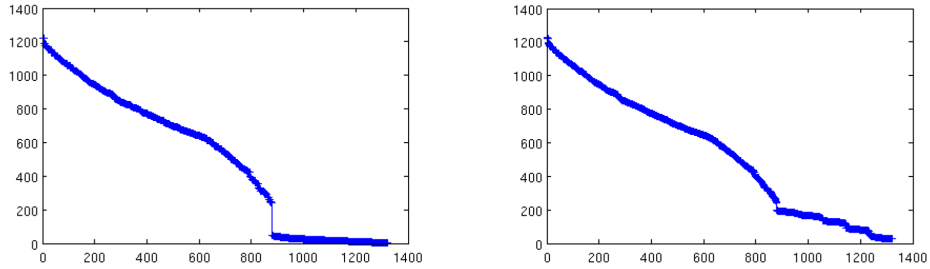


Figure 4.3: Singular values for a 5 frames sequence under perspective projection. Left: Without temporal consistency constraints between frames, the linear system is ill-constrained. Right: Bounding the frame-to-frame displacements along the line of sight using the linear expression of Eq. 4.4 transforms the ill-conditioned linear system into a well-conditioned one. The smaller singular values have increased and are now clearly non-zero. Since our motion model introduces more equations than strictly necessary, the other values are also affected, but only very slightly.

is designed to give comparable influence to both classes of constraints and directly affects how much the small singular values increase.

4.2 Experimental Results

In the previous sections, we developed theoretical basis for reconstructing the shape of a deformable surface from 3D-to-2D correspondences in a video sequence. We showed that constraining the variations in depth from frame to frame is sufficient, in theory, to formulate the reconstruction problem in terms of solving a well-conditioned linear system. In this section, we show that this indeed produces valid reconstructions in practice.

We present results obtained using both synthetic data and real images. In both cases, the deformations of the meshes were retrieved by solving the linear system of Eq. 4.5 for whole sequences with known deformations in the first and last frames. This was done using Matlab's implementation of sparse matrices and resolution of linear systems with known covariance matrix in the least square sense. In our experiments, the covariance matrix simply is the weight matrix of Eq. 4.6, which weighs differently the correspondences equations and the constraints. Additionally, we assumed that the shape of the surface in the last frame of the sequence is also known, thus adding an extra set of constraints similar to those linking the first and second frames. This nonetheless is essentially aesthetic and is not a true requirement of our approach.

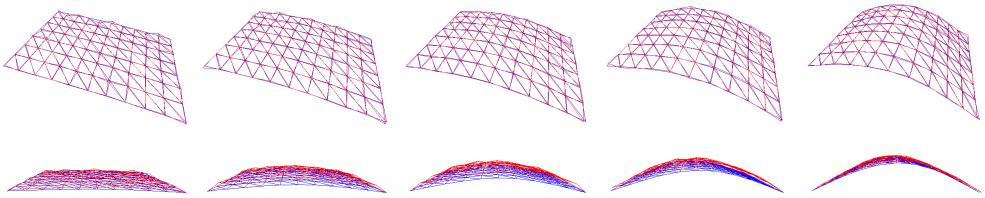


Figure 4.4: Reconstructing an 88-vertices mesh using perfect correspondences that were corrupted using zero-mean Gaussian noise with variance five, which is much larger than what can be expected of automated matching techniques. Top: The original mesh and reconstructed one projected in the synthetic view used to create the correspondences. As expected, the projections match very closely. Bottom: The two meshes seen from a different viewpoint.

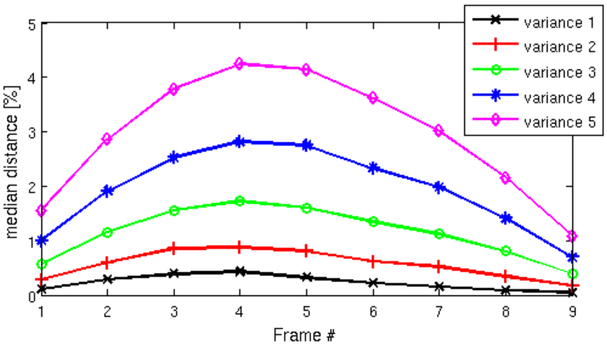


Figure 4.5: Distance between the original mesh and its reconstruction for each one of the 9 deformed versions of the mesh of Fig. 4.4. We plot five curves corresponding to vertex-to-surface distances obtained with variance one to five gaussian noise on the correspondences. The distances are expressed as percentages of the length of the mesh largest side. Note that knowing the final shape avoids having a monotonically increasing error.

4.2.1 Synthetic Data

We deformed the 88-vertex mesh of Fig. 3.3(a) to produce 9 different shapes and 9 corresponding sets of 3D-to-2D correspondences using a perspective projection matrix. We then added Gaussian noise with mean zero and variance ranging from one to five to the image locations of these correspondences. Fig. 4.4 depicts the reconstruction results overlaid on the original mesh with noise variance five. The differences are hard to see, even though this represents far lower precision than what can be expected of good feature point matching algorithms.

To quantify the differences between the meshes, we plot the distances between the two meshes in Fig. 4.5 for each one of 9 different shapes, given increasing noise variance. The distances are expressed as percentages of the mesh largest side. With a noise variance one,

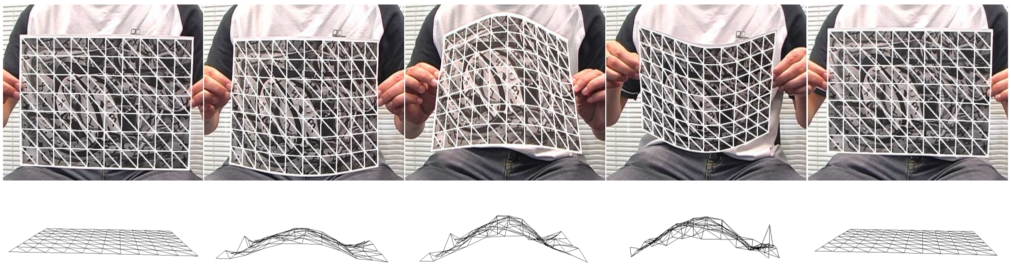


Figure 4.6: Reconstructing a deforming sheet of a paper from a 250-frames sequence. Top: The reconstructed mesh is reprojected into the original images and closely matches the outline of the paper. Bottom: The same mesh seen from the side. In spite of local inaccuracies in depth, the overall shape is correct, which indicates that the ambiguities have been successfully resolved.

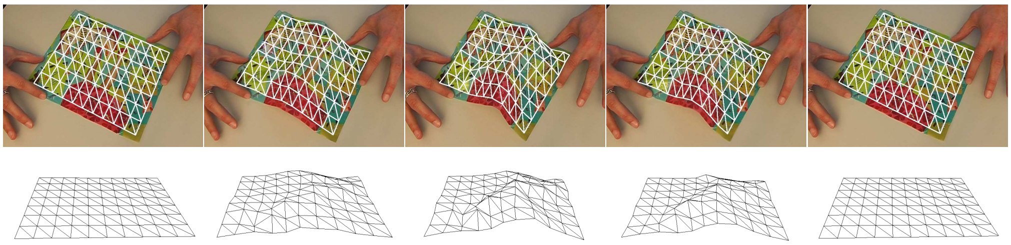


Figure 4.7: Reconstruction results for a plastic sheet, which is much more flexible than the sheet of paper of Fig. 4.6. In spite of this, the overall shape is again correctly recovered up to small errors due to erroneous correspondences.

they are of the order of 0.25% for vertex-to-surface distance, which works out to 0.025cm for a 10cm×7cm mesh. This is very small given that we incorporate very little *a priori* knowledge into our reconstruction algorithm.

4.2.2 Real Data

We now present results on two real monocular video sequences acquired with an ordinary digital camera. The longest one is 250 frames long, which shows that, even though our approach involves solving a very large system, it is sparse enough to use a standard Matlab routine. In both cases, we automatically establish 3D-to-2D correspondences between the first frame, where the 3D pose is assumed to be known, and the others by first tracking the surface in 2D and then computing a dense matching as explained in Section 3.3.1.2. This results in noisy correspondences with a number of mismatches at places where there is not enough texture to guarantee reliable matches.

Fig. 4.6 depicts our reconstruction results for a relatively inelastic piece of paper in a 250-frames sequence and Fig. 4.7 those for a much more flexible sheet of plastic in a 147-frames sequence. In both cases, the global shape is correct, which confirms that the ambiguities have been correctly handled. However, because we impose no smoothness constraint of any kind, there are also local errors that are caused by the mismatches present in our input data. If the goal were to derive a perfect shape from a set of noisy correspondences, we could mitigate the effect of erroneous matches by introducing a robust estimator into the least-squares minimization of Eq. 4.6.

Since our technique does not introduce any prior on the physical properties of the target surface, we were able to reconstruct both the paper and plastic without changing anything to our system.

4.3 Convex Optimization

Even though the constraints introduced in the previous section exactly address the ambiguities of our problem, they almost never accurately model the true dynamical behavior of a non-rigid surface. Indeed, these constraints enforce the motion along the line of sight to be zero, which is only true when the surface does not deform. Another drawback of the previous approach is that the linear formulation of the correspondence problem does not exactly minimize the reprojection error. It is equivalent to applying a Direct Linear Transformation (DLT) [60], since the reprojection of a 3D point on the image plane would involve a division by the depth factor k, thus yielding non-linear terms, as can be checked in Eq. 3.1. Instead of the distance in the image plane between the projected 3D point and its corresponding measurement, our linear formulation describes an error at some depth, which varies from one correspondence to another and makes them influence the global error differently. The true reprojection error is a non-linear function of the 3D point coordinates, and thus cannot fit our linear formulation.

It was recently shown that several computer vision problems such as triangulation, camera resectioning and homography estimation can be formulated as convex optimization problems [82, 85]. Such formulations involve solving an optimization problem for which the objective function and the constraints are convex. Linear programming (LP), Second Order Cone Programming (SOCP) and Semi-Definite Programming (SDP) are examples of classes of convex optimization problems. Some Quadratic Programming (QP) problems and Quadratically Constrained Quadratic Programming (QCQP) problems can also be written as convex optimization problems. Finally, quasi-convex optimization is a relaxation of convex optimization where the function is not strictly convex, but has a single minimum over a convex domain. More details on convex optimization can be found in [21].

In the various computer vision applications described in [82, 85], the correspondence problem is formulated as a Second Order Cone Programming feasibility problem. This is a particular type of convex optimization problems where no function is minimized. Instead, one looks for a vector $\mathbf{X}$ that satisfies the m constraints

$$\|\mathbf{A_i}\mathbf{X} + \mathbf{b_i}\|_2 \leq (\mathbf{c_i}^{\mathrm{T}}\mathbf{X} + d_i)\ ,\ \text{for } i = 1, ..., m\ . \tag{4.7}$$

Such problems can be solved very effectively using available packages such as SeDuMi [147].

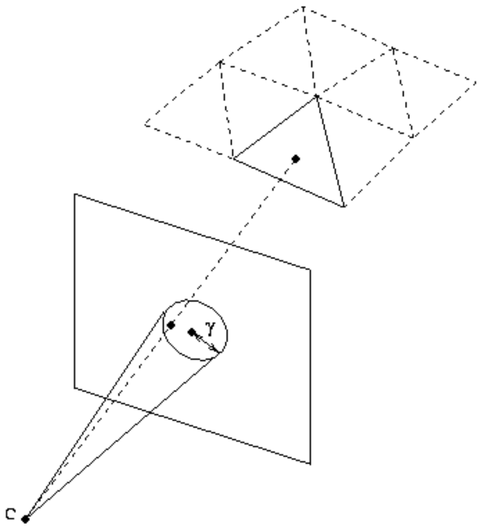


Figure 4.8: A cone of radius γ is defined for each correspondence. It is centered in the camera, and its axis goes through the image measurement. 3D points on the surface must reproject inside their corresponding cone on the image plane.

In the particular context of reprojection error, it has been shown that minimizing the $L_\infty-\mathrm{norm}$ could be done by iteratively solving such an SOCP feasibility problem. In other words, given a set of correspondences, one can find the solution that minimizes the largest reprojection error. The drawback of this formulation is its sensitivity to outliers, since a single mismatch would yield a high maximum reprojection error, thus allowing large variations of the solution. Fortunately, a solution was proposed in [141]. However, such a quasi-convex formulation has only been demonstrated for rigid objects. Here, we extend this approach to deformable 3D surfaces.

4.4 SOCP for Deformable Surfaces

In this section, we show that recovering the 3D shape of a deformable surface from a single video sequence can be formulated as an SOCP problem, as described by Eq. 4.7. As in our earlier linear formulation, we obtain dense 3D-to-2D correspondences using the technique described in Section 3.3.1.2, and assume that the camera projection matrix $\mathbf{P}$ is known and remains constant. This does not mean that the camera cannot move, but that we can only recover a relative motion of the surface with respect to it.

4.4.1 Correspondences as SOCP Constraints

As in the linear case, we can consider a 3D point $\mathbf{q}_i$ defined by its barycentric coordinates $(a_i,\ b_i,\ c_i)$, and its corresponding image measurement $[u_i\ v_i]^T$. The projection of $\mathbf{h}_i =$

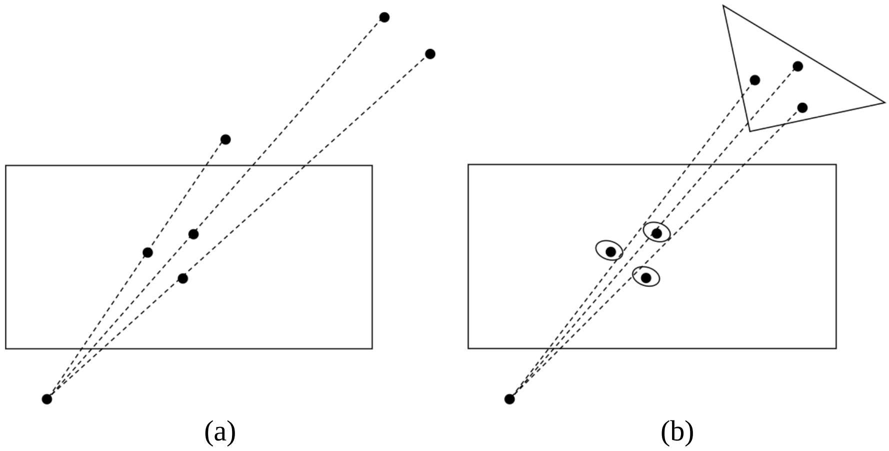


Figure 4.9: Using a mesh imposes constraints on the reconstruction. (a) Without a mesh, nothing would prevent the 3D points to perfectly match their corresponding image locations. However, this would yield a completely meaningless shape. (b) The barycentric coordinates link several 3D points together, and thus impose a natural coherence between them.

$\left[\mathbf{q}_i^T \; 1\right]^T$ given the camera projection matrix $\mathbf{P}$ is

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{P}_1\mathbf{h}_i}{\mathbf{P}_3\mathbf{h}_i} \\ \frac{\mathbf{P}_2\mathbf{h}_i}{\mathbf{P}_3\mathbf{h}_i} \end{bmatrix} ,$$

where $\mathbf{P}_k$ refers to the k^{th} row of the projection matrix. We define the reprojection error in the image plane as

$$\left\| \frac{\mathbf{P}_1\mathbf{h}_i}{\mathbf{P}_3\mathbf{h}_i} - u_i, \frac{\mathbf{P}_2\mathbf{h}_i}{\mathbf{P}_3\mathbf{h}_i} - v_i \right\| = \frac{\|(\mathbf{P}_1 - u_i\mathbf{P}_3)\mathbf{h}_i, (\mathbf{P}_2 - v_i\mathbf{P}_3)\mathbf{h}_i\|}{\mathbf{P}_3\mathbf{h}_i} . \tag{4.8}$$

Note that this, as opposed to the linear formulation, is the true reprojection error, which is a non-linear function of the mesh vertices.

Ideally, we would want the reprojection error to be zero for all $\mathbf{q}_i$, $1 \leq i \leq N_c$ for which we have found a corresponding image point (u_i, v_i). In practice, due to noise, this is never possible. Therefore, as in [82], we introduce an additional variable γ and write our problem as

$$\begin{aligned} \min_{\gamma,\mathbf{y}} \gamma \;\; & \text{subject to } \gamma \geq 0 \text{ and} \\ & \|(\mathbf{P}_1 - u_i\mathbf{P}_3)\mathbf{h}_i, (\mathbf{P}_2 - v_i\mathbf{P}_3)\mathbf{h}_i\| \leq \gamma\mathbf{P}_3\mathbf{h}_i \; , \\ & \text{for } i = 1, ..., N_c \; . \end{aligned} \tag{4.9}$$

where $\mathbf{y}$ is the concatenation of the three coordinates of all the mesh vertices. Intuitively, γ represents the radius on the image plane of the cone centered in the camera and whose

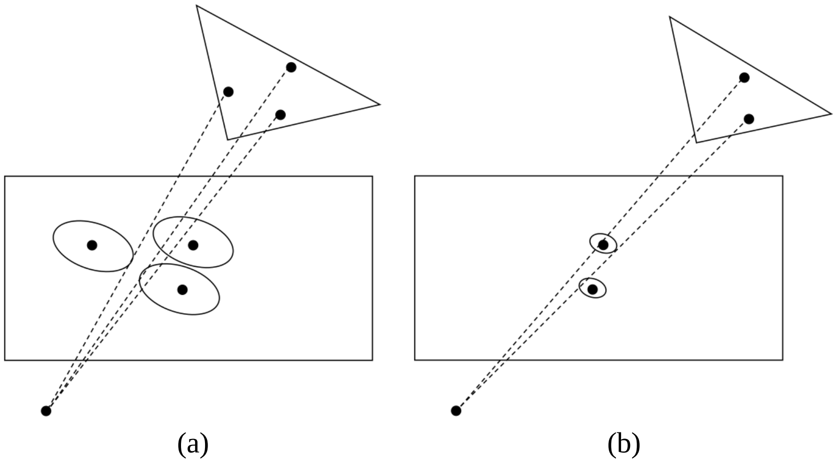


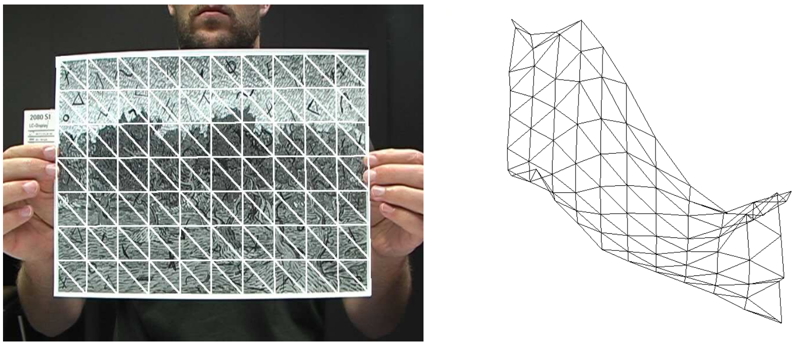
Figure 4.10: The SOCP formulation of the correspondence problems is very sensitive to outliers, since it minimizes the L_∞-norm. (a) With a single outlier, the minimum cone radius γ remains very large, and thus allows the correct matches to reproject far from their corresponding image locations. (b) Once the outlier is removed, γ can take a much smaller value, which yields a much better shape.

axis goes through the image measurement, as depicted in Fig. 4.8. A single γ is used for all the correspondences, thus enforcing all the reprojection errors to be less than γ.

For a fixed value of γ, because the $\mathbf{q}_i$, and therefore also the $\mathbf{h}_i$ are linear combinations of the vertex coordinates, Eq. 4.9 defines an SOCP feasibility problem as described by Eq. 4.7. We can then find the minimal γ using a bisection algorithm at each step of which we solve the corresponding SOCP feasibility problem. This yields reprojection errors that are all smaller than the minimum γ found, and therefore minimizes their L_∞-norm.

Note that the fact that the 3D points are on the surface of a mesh plays a critical role, as illustrated by Fig. 4.9. Without this constraint, they could move independently from each other. Since nothing would then prevent them to match their 2D projection within a zero-radius cone, this would result in a perfect but meaningless solution. However, forcing them to remain on the surface of the deformable mesh avoids that problem, since the barycentric coordinates that define the 3D points impose a natural coherence between them.

A common criticism of SOCP formulations of the correspondence problem is that they are very sensitive to outliers. Indeed, the minimal γ will take the value of the worst correspondence, therefore allowing the reprojection errors of correct matches to be worse than they should, as shown in Fig. 4.10. However, Sim et al. [141] proposed a method to remove the outliers and get the correct pose of a rigid object using an SOCP approach. They showed that, at the end of the bisection algorithm, the set of matches whose reprojection error equals the minimal γ contains outliers. Therefore, removing these points and re-optimizing in the same manner as before yields a better pose. In our implementation, we apply the same idea and iterate the bisection algorithm with the correspondences having a



(a) (b)

Figure 4.11: Reconstructing a piece of paper using only the correspondences constraints of Section 4.4.1 but not the deformation constraints of Section 4.4.2. (a) The reprojection of the mesh is correct. (b) However, the 3D shape as seen from a side view is completely wrong because the depth ambiguities are not properly resolved.

reprojection error less than the previous minimal γ, until we reach a maximal reprojection error of 2 pixels. In practice, this only implies running the bisection algorithm at most 5 times.

4.4.2 Additional Constraints

In general, solving the minimization problem of Eq. 4.9 without additional constraints yields a surface whose points project at the right place but whose overall shape may nevertheless be wrong, as shown in Fig. 4.11. As shown in Chapter 3, the global scale of the surface can vary without affecting the reprojection error, and, more damagingly, given noisy data, the depth of the vertices is hard to precisely estimate because many different shapes can yield very similar projections.

At the begining of this chapter, we introduced a linear motion model that precisely reduced these ambiguities. However, the particular form of the constraints never truly models the real dynamic behavior of a surface, since it assumes that the depths of the vertices remain unchanged. Another approach to address this problem is the use of penalty functions. They are usually designed either to prevent the mesh from folding sharply or to stop it from expanding or shrinking. The former results in a loss of generality as surfaces that crease cannot be modeled properly, while the latter typically involves a non-convex term to force the edges of the mesh to retain their original length.

Here, we introduce a weaker and more generic motion model that fits into our SOCP framework: As shown in Fig. 4.12, we avoid the orientation of the edges to change irrationally between two consecutive frames. In the meantime, our constraints also ensure that an edge will not stretch or compress too much, thus partially solving the global scale ambiguity. Independently of the surface's curvature, this is generally applicable when tracking

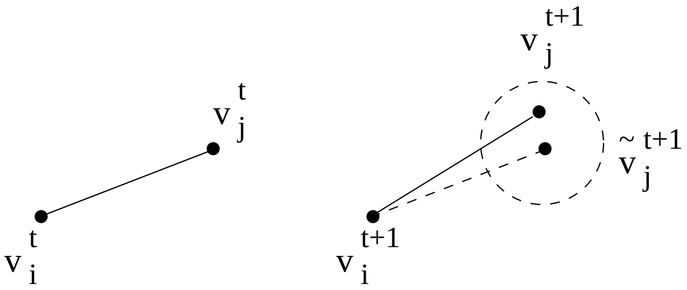


Figure 4.12: We predict that the orientation of the edge between $\mathbf{v}_i$ and $\mathbf{v}_j$ at time $t+1$ will be the same as at time t. We then constrain the distance between vertex $\mathbf{v}_j^{t+1}$ and its prediction $\tilde{\mathbf{v}}_j^{t+1}$ to be less than some specified value.

it in a 25 frames-per-second video sequence. Furthermore, it can be expressed as a convex constraint as follows.

Let us assume that we know the shape of the mesh at time t and let us consider an edge linking vertices $\mathbf{v}_i^t$ and $\mathbf{v}_j^t$ in this configuration. Assuming that the orientation of this edge will be similar at time $t+1$, if the position $\mathbf{v}_i^{t+1}$ of vertex i at that time were known, we could predict that of vertex j to be close to

$$\tilde{\mathbf{v}}_j^{t+1} = \mathbf{v}_i^{t+1} + L_{i,j} \frac{\mathbf{v}_j^t - \mathbf{v}_i^t}{\|\mathbf{v}_j^t - \mathbf{v}_i^t\|} \ , \tag{4.10}$$

where $L_{i,j}$ is the original length of the edge. In practice we do not know $\mathbf{v}_i^{t+1}$ but we can nevertheless require that

$$\|\mathbf{v}_j^{t+1} - \tilde{\mathbf{v}}_j^{t+1}\| \leq \lambda L_{i,j} \ , \tag{4.11}$$

where $\tilde{\mathbf{v}}_j^{t+1}$ is defined in Eq. 4.10. This constraint fits perfectly within our SOCP framework and we can add one for each edge to those of Eq. 4.9. Note that these additional constraints allow the mesh to expand or shrink, but only within an amount controlled by the user-defined value λ. Given a mesh that satisfies the SOCP constraints, we handle the remaining scale ambiguity by rescaling it so that its area remains the same as in the initial position. As will be shown in Section 4.5, this results in a system that is now sufficiently constrained to yield good reconstructions.

4.5 Experimental Results

In the previous section, we showed how convex optimization can be applied to the problem of recovering the 3D shape of a surface from a single video sequence. We presented constraints that do not prevent the surface from folding sharply, by making assumptions on the frame-to-frame deformations of the object. Our method relies on 3D-to-2D correspondences and only requires the pose in the first frame of the sequence to be known.

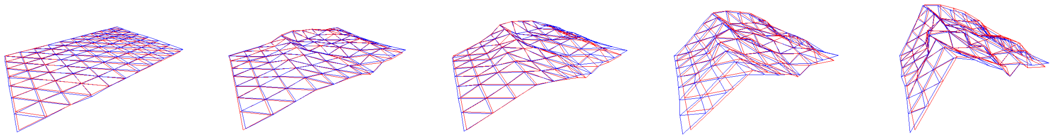


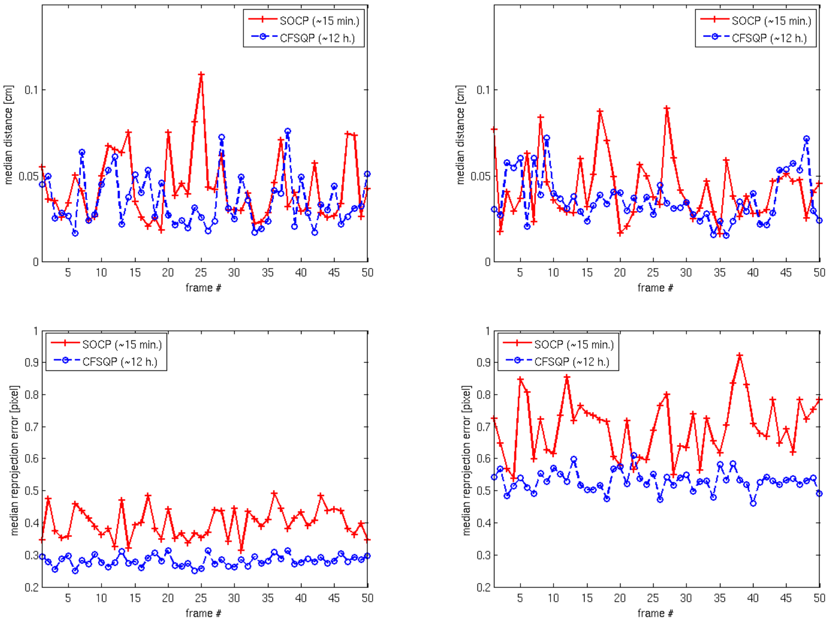
Figure 4.13: Reconstructing an 88-vertices mesh with sharp folds using perfect correspondences that were corrupted using zero-mean Gaussian noise with variance two. The shape of the reconstructed mesh (blue) corresponds very closely to the original one (red). The meshes are seen from a different perspective than the one used to retrieve the shapes in order to highlight the differences.

We first validate our approach using synthetic data. We then use ordinary videos to demonstrate that it produces good results for very different kinds of materials. In all our experiments, both for synthetic and real data, the value of λ in Eq. 4.11 was set to 0.1, independently of the properties of the surface and of its deformations.

4.5.1 Synthetic Data

As a first experiment, we synthetically deformed the 88-vertices mesh shown in Fig. 4.13 by applying forces to randomly chosen vertices and strongly penalizing stretching of its edges. This produced a sequence of 50 different shapes, from which we could obtain correspondences by projecting 3D points defined by their randomly chosen barycentric coordinates using a perspective projection matrix. We then added gaussian noise with mean zero and variance one and two to their image locations. Fig. 4.13 shows the reconstruction results for variance two from a different perspective. The differences are very small even though the surface folds very sharply in some frames. The largest errors are in the depth direction, as could be expected since motion in that direction is hard to measure using point correspondences. The deformation constraints of Section 4.4.2 resolve most of the resulting ambiguities but still leave some uncertainty.

To compare the performance of SOCP against another powerful optimization technique, we reimplemented our tracking algorithm using CFSQP [93], which provides C functions to solve constrained minimization problems using Sequential Quadratic Programming. We reformulated our problem as the minimization of the sum of squared reprojection errors under the deformation constraints of Section 4.4.2. The top row of Fig. 4.14 shows the median vertex-to-ground-truth-surface distances for noise variances 1 and 2 for each one of the frames in the synthetic sequence. The distances are of the order of 0.1cm for a mesh of size 10cm×7cm. Both the SOCP and CFSQP implementations produced roughly comparable errors. However, even though SOCP was coded in Matlab whereas CFSQP was coded in C, SOCP was about 50 times faster than CFSQP: It took only 15 minutes against 12 hours to process the whole sequence on the same 3.0 GHz PC. The second row of Fig. 4.14 shows the median reprojection errors over the correspondences. CFSQP yields slightly higher accuracy, but the errors still remain under one pixel for SOCP. This can



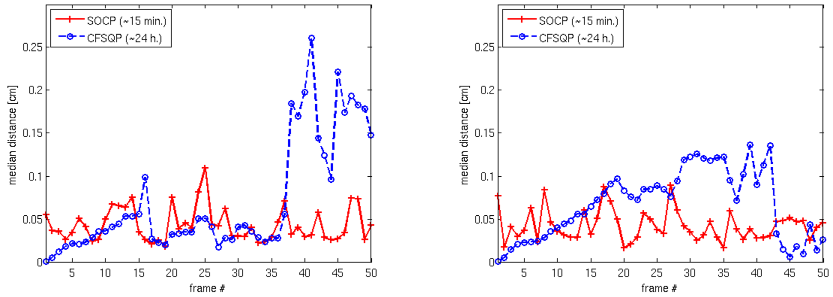
$\sigma = 1$ $\sigma = 2$

Figure 4.14: We compare the results of our SOCP formulation (solid red) against those obtained using CFSQP, a constrained non-linear least-squares minimization (dashed blue) for the 50 frames of the synthetic sequence of Fig. 4.13, for noise variance $\sigma = 1$ and 2. In the top row, we show the distance between the original mesh and its reconstruction. Both methods give similar results but SOCP is about 50 times faster. In the second row, we give the median reprojection errors. For both methods, they are less than one pixel, even though CFSQP performs slightly better. Recall, however, that SOCP does not precisely minimize the reprojection errors, but enforces the reprojections to lie in a cone of a given radius.

be explained by the fact that SOCP does not minimize the reprojection errors, but finds a solution such that these errors are smaller than a given value.

Since CFSQP can handle non-convex constraints, we replaced the deformation constraints of Section 4.4.2 by constraints that prevent the mesh edges from changing their length. In theory, this should be more appropriate when tracking inextensible surfaces. In practice, as shown in Fig. 4.15, even though CFSQP performs better in some frames, it is less stable than SOCP. This is particularly visible towards the end of the sequence. In some frames, CFSQP failed to converge even after 2000 iterations, which explains why it is even slower than before and highlights the complexity of the problem when non-convex constraints are used.

In Fig. 4.16, we show the influence of the number of correspondences on the quality of



$\sigma = 1$ $\sigma = 2$

Figure 4.15: Introducing non-convex inextensibility constraints, for noise variance $\sigma = 1$ and 2. Since CFSQP can handle such constraints, we introduce them into our CFSQP formulation and, as in Fig. 4.14, compare the results (in blue) against those of SOCP (in red). In addition to being much slower, CFSQP gives unstable results and fails to converge in some frames after 2000 iterations.

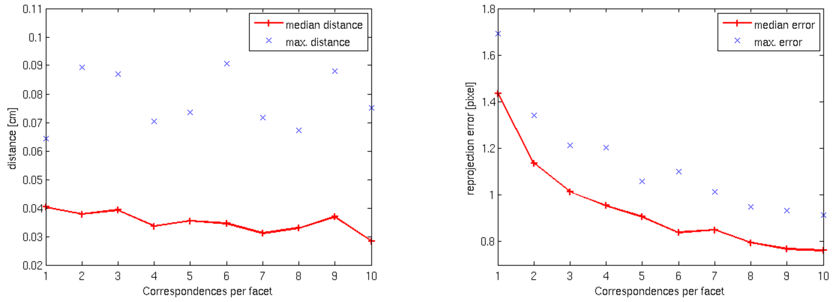


Figure 4.16: Influence of the number of correspondences in each facet on the reconstruction for noise variance $\sigma = 2$. We decreased the number of correspondences per facet from 10 to 1, and display the median (red line) and maximum (blue crosses) values of the same errors as in Fig. 4.14. We show the 3D distance errors in the left image, and the reprojection errors computed for all 10 correspondences per facet in the right one. Note that the 3D vertex-to-surface distance is little affected by the correspondences, whereas the reprojection error decreases in a more noticeable manner, which is to be expected, since, in the first cases, not all correspondences were used during optimization.

our reconstruction in the case of a variance 2 gaussian noise. We decreased the number of correspondences in each facet from 10 to 1 and tracked the surface throughout the 50 frames of the sequence. For each frame, we computed the median vertex-to-ground-truth-surface distance and median reprojection error for all 10 correspondences per facet. For each number of correspondences per facet, we display on the left image, the median and maximum values of such 3D distances over the sequence, and on the right image, the

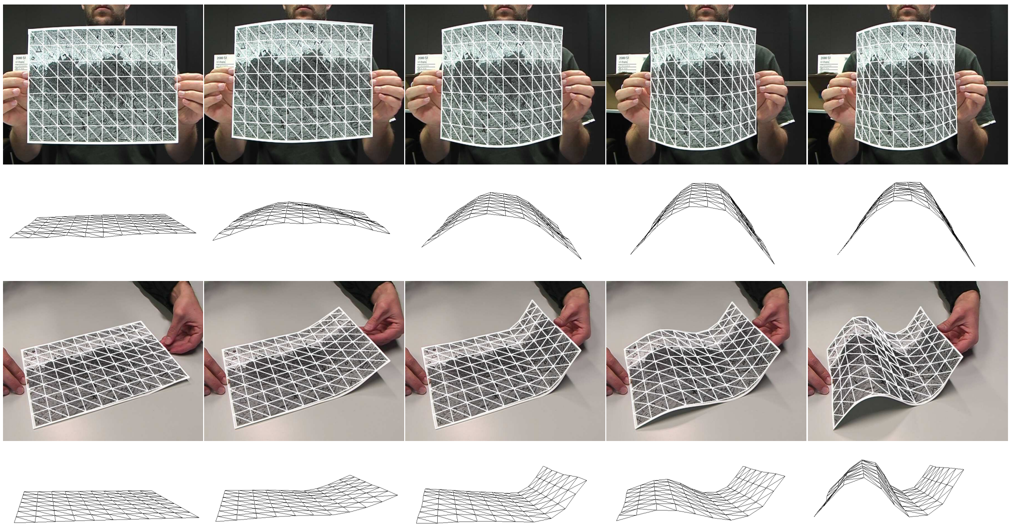


Figure 4.17: Reconstructing a deforming sheet of paper from a 71 frames video and a 116 frames video. The mesh is reprojected in the image in the top row and seen from a different perspective in the bottom one. Even though no smoothness constraint was enforced, the algorithm correctly recovered smooth deformations.

median and maximum values of such reprojection errors over the sequence. The number of correspondences has little influence on the 3D distances, since the vertices can slide along the true surface without changing these measures. The reprojection errors are more strongly affected, but note that, from 4 correspondences per facet, they drop below one pixel. Additionally, for cases with less than 10 correspondences per facet, the reprojection error is affected by matches that were not taken into account during optimization, and that therefore tend to increase the error. Of course, this still assumes at least one correspondence in each facet. With such a weak deformation model, if some facets did not contain any, the reconstruction would inevitably degrade. This would especially be the case for facets on the boundary of the surface, since their vertices are constrained by fewer neighbors than the ones in the middle.

Finally, we compared the results of the SOCP motion model with those obtained with the linear motion model introduced in Section 4.1.2. For this purpose, we used data reconstructed from an optical motion capture system, as explained in more details in Chapter 6. The results of this comparison are shown in Figs. 7.11 and 7.13 of Chapter 7, where we compare all our methods together. From these plots, it is obvious that the SOCP formulation performs better than the linear one. This is not surprising since the linear constraints were only designed to be the minimal ones that address our ambiguities, and because SOCP truly models the reprojection error.

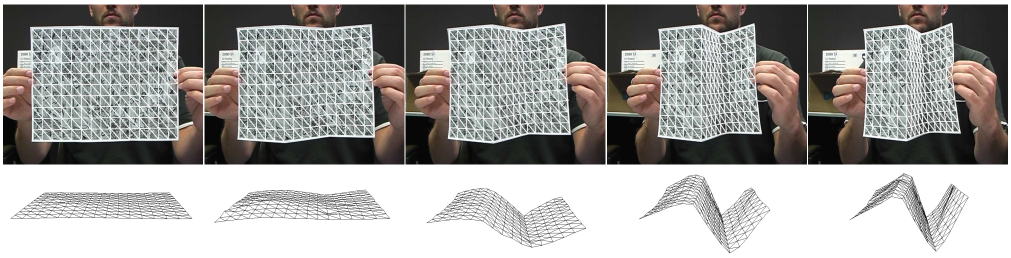


Figure 4.18: Reconstructing the deformations of a piece of paper with two sharp folds in it, so that that they are no longer smooth. Note that our method correctly recovers the creases.

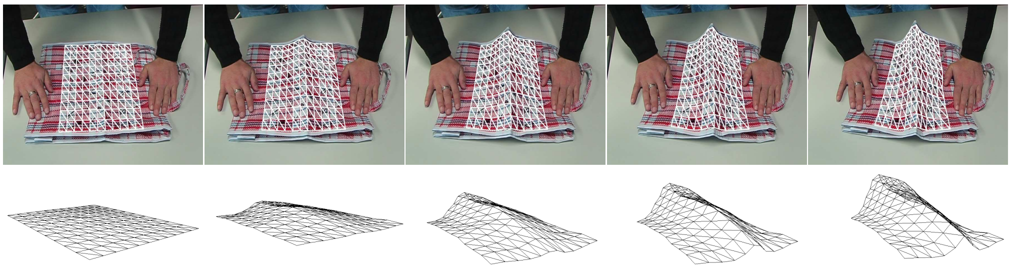


Figure 4.19: Recovering the deformations of a plastic bag with a sharp crease in it from from an 86 frames video.

4.5.2 Real Data

We now show reconstruction results of real deformable surfaces made of paper, cloth, and plastic. The video sequences were acquired with an ordinary digital camera. Due to their very different physical properties, the behavior of the surfaces ranges from smooth deformations for the paper to sharp folds and creases for the cloth and plastic. However, no parameter tuning was necessary to obtain these results with our algorithm.

Smooth Deformations As a first experiment on real data, we considered a sheet of paper that we modeled as an 88-vertex mesh. Fig. 4.17 shows that we can retrieve the correct shape of a surface that deforms smoothly, even though our formulation involves no penalty term on the curvature of the reconstructed surface.

Sharper Folds Because we do not penalize curvature, nothing stops our method from recovering the correct shape in the presence of folds and creases, as demonstrated by the reconstruction of the pre-folded sheet of paper of Fig. 4.18, the plastic bag of Figs. 4.19 and 4.20, and the piece of cloth of Figs. 4.21, 4.22 and 4.23.

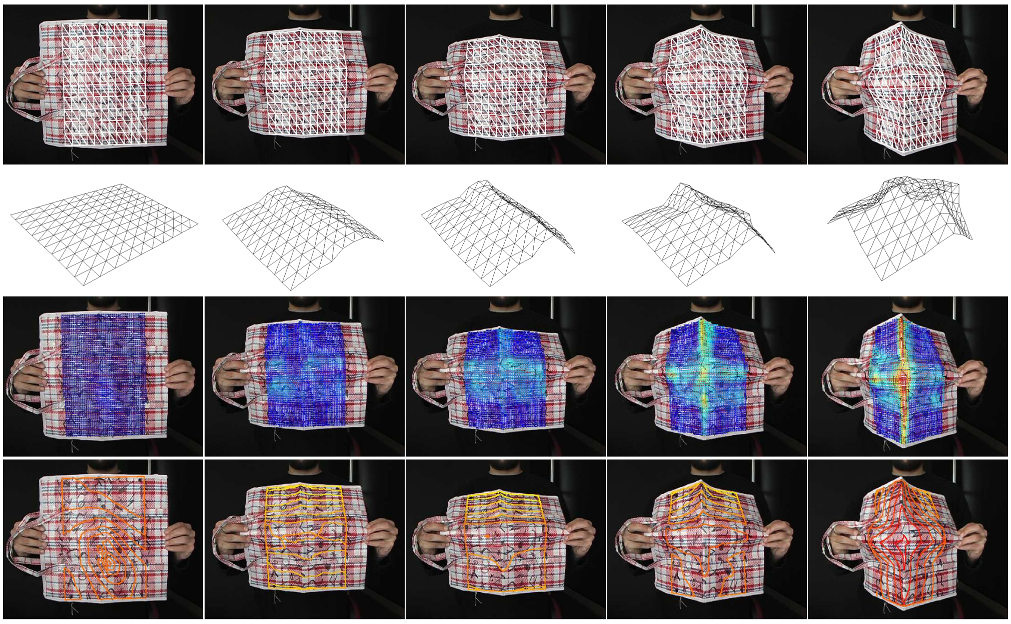


Figure 4.20: Recovering more complex deformations of the plastic bag. The first two rows depict the reprojection of the mesh into the original images and the mesh seen from a different perspective as before. In the third row, we overlay the mean curvature of the recovered surface on the images. The high curvature areas, shown in red, correspond to the actual creases that can be seen in the top row. In the fourth row, we overlay the level-lines of constant z on the images. We recommend viewing the last two rows in color as they might be difficult to interpret on a greyscale printed copy.

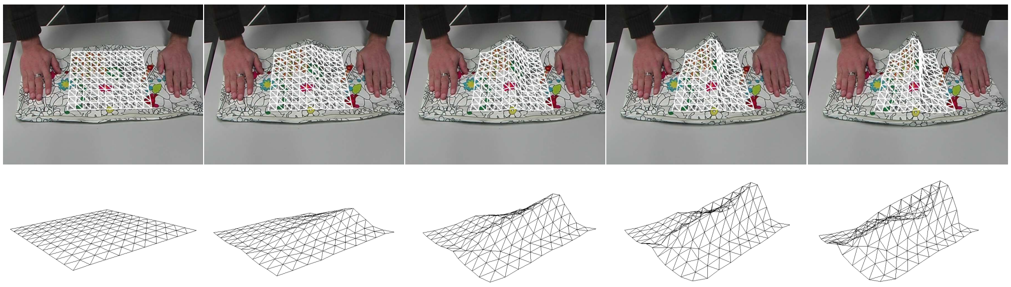


Figure 4.21: Recovering the deformations of a piece of cloth from a 50 frames video.

4.6 Conclusion

In this chapter, we introduced two different motion models to overcome the ambiguities of monocular reconstruction that were formalized in the previous chapter. The first model is linear and penalizes depth motion from one frame to the next. It directly addresses the

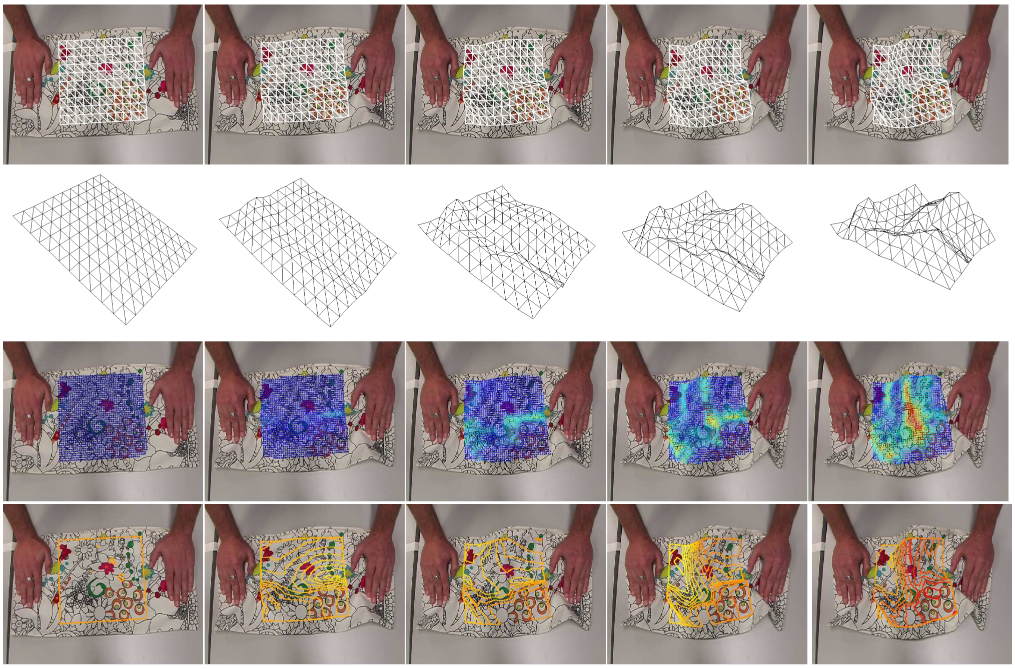


Figure 4.22: Recovering the deformations of a piece of cloth with several folds. The third and fourth rows depict the same curvature and level-line information as in Fig. 4.20 and are best viewed in color.

observed ambiguities, and can therefore be considered as the minimal set of constraints required for monocular reconstruction. However, being minimal, these constraints typically never hold for real sequences and make a poor prediction of the surface shape. Furthermore, formulating the correspondence problem as the solution to a linear system only approximates the true reprojection errors and leads to treating the correspondences unequally.

We then replaced our initial linear formulation of the correspondence problem by a convex optimization one that correctly models the reprojection errors. This let us define more appropriate motion constraints that penalize excessive frame-to-frame changes of edge orientation, as well as excessive edge length variations over the whole sequence. This yielded results that are both accurate and less sensitive to noise than with the previous method. Furthermore, it proved effective to retrieve the shape of smoothly deforming surfaces as well as surfaces undergoing complex folds and creases.

Nonetheless, this formulation still has weaknesses. First, it requires the presence of correspondences over the whole surface, because such generic constraints are not sufficient to interpolate the shape of untextured parts from the textured ones. Additionally, motion models have the disadvantage to link results from one frame to the next, therefore increasing the risk of accumulating reconstruction errors throughout the sequence. Finally, this representation uses all the vertex coordinates as unknowns, whereas is it widely accepted that the possible deformations of a surface lie on a much lower dimensional manifold. Rather than

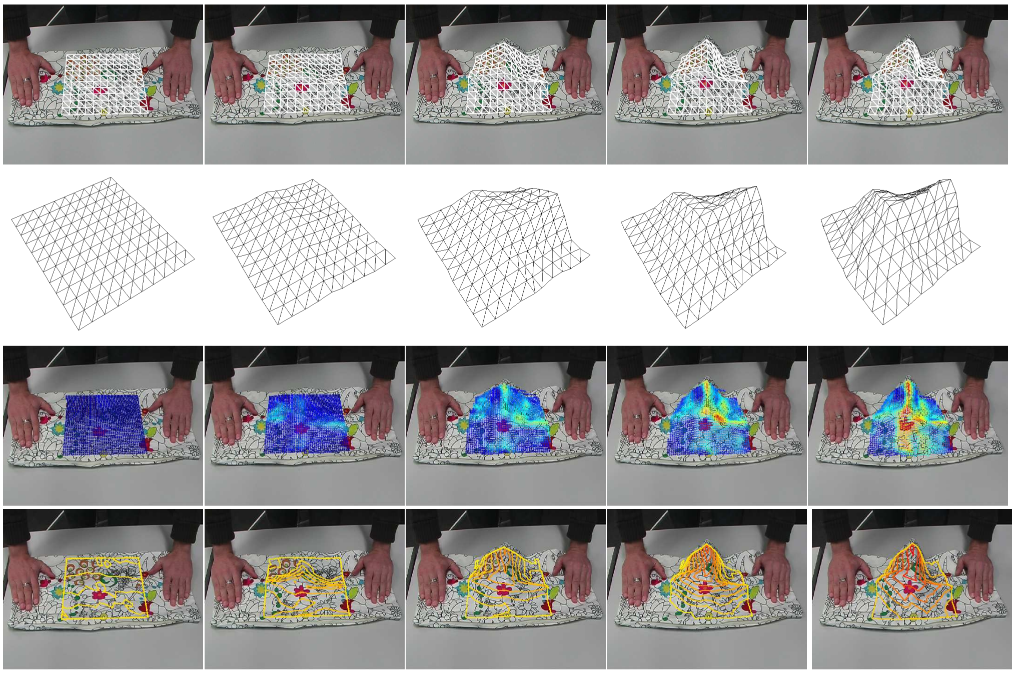


Figure 4.23: Another example of a different deformation of that same cloth in a 61 frames sequence. The third and fourth rows depict the same curvature and level-line information as in Fig. 4.20 and are best viewed in color.

explicitly adding constraints that enforce the resulting shapes to remain on this manifold, it would be more efficient to use parameters that implicitly encode these constraints. This is what we propose to do in the upcoming chapters, by introducing deformation models learned from training examples.

5 Global Deformation Models

In this chapter, we introduce global deformation models that are learned from training data. Whereas so far we have parameterized the global shape of a surface in terms of the vertex coordinates of its mesh representation, we now model it as a linear combination of modes, or basis shapes, which involves far fewer parameters. The methods presented in this chapter are directly inspired by Active Appearance Models [37], Morphable Models [19] and Structure-from-Motion approaches [25, 24, 154], which we discussed in Section 2.2.

A well-known issue with such representations is that they depend on training examples that can be very hard to obtain and even harder to register together. Even though this has been done for faces [19], and is therefore feasible, it involved a painstaking manual process which would be even harder for generic deformable surfaces. In structure-from-motion [24, 156, 154], this has been addressed by learning the bases online from the input video. However, this requires a large number of images, and produces bases that only describe the particular deformations observed in that sequence. Here, we rather propose to use modes learned offline from training examples. We consider the case of inextensible surfaces, and propose a way of automatically creating deformed versions of a single topology, thus directly yielding registered shapes. From these shapes, we compute deformation modes and use them to reconstruct surfaces from monocular videos. Even though our basis shapes were constructed from inextensible surfaces, we show that they let us reconstruct extensible ones.

Finally, in this framework, we show that reconstruction can be done in closed-form given a single input image. Following the same linear formulation of the correspondence problem as before, we overcome the depths ambiguities by introducing quadratic equations accounting for inextensibility and solving the resulting system. This alleviates the need of a video sequence, and lets us recover the shape of surfaces from individual images.

5.1 Inextensible Triangulations

One of the simplest ways to model a deformable surface is to represent it as a triangulated mesh parameterized in terms of its vertex coordinates. This parameterization, however, does not account for the fact that, in a real surface, the vertices cannot move independently from one another. By contrast, if we constrain the triangulation edges to retain their original length, the number of degrees of freedom (dofs) decreases very significantly. At first sight, considering inextensible surfaces only can seem very restrictive. However, at the level of details that a standard camera can capture, many ojects, such as sheets of paper, clothes, or sails, are inextensible. Furthermore, as we will show later, our final representation will still

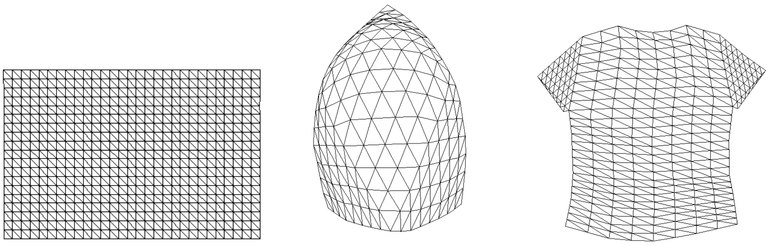

(a) (b) (c)

Figure 5.1: Hexagonal triangulations. (a) Rectangular mesh used to model a piece of paper. (b) Triangular mesh used to model a spinnaker. (c) Stitching a rectangular patch for the body part and two triangular ones for the sleeves lets us model a t-shirt.

have the ability to model stretchable surfaces, and therefore lacks no generality.

5.1.1 Dofs of Inextensible Triangulations

We seek to characterize the number of dofs of a triangulation—containing N_v 3D vertices, N_f facets, and N_e edges—that has a planar topology, which means it can be unfolded to a plane and has an actual boundary that can form an arbitrary polygon. In general, such a triangulation has 3 dofs per vertex. However, forcing the edges to retain their length when the triangulation deforms, imposes one quadratic constraint per edge and the total number of degrees of freedom drops to

$$N_d = 3N_v - N_e \ . \tag{5.1}$$

Let N_e^b be its number of boundary edges and $N_e^i = N_e - N_e^b$ the number of interior ones. Since the N_e^b boundary edges each belong to only one facet whereas the N_e^i internal ones belong to two, we have

$$3N_f = 2N_e^i + N_e^b \ . \tag{5.2}$$

Furthermore, according to Euler's well known formula, if the triangulation has no holes,

$$N_v + N_f - N_e = 1 \ . \tag{5.3}$$

Substituting Eqs. 5.2 and 5.3 into Eq. 5.1 yields

$$N_d = 3 + N_e^b \ \ . \tag{5.4}$$

In other words, the number of degrees of freedom of an inextensible triangulation grows as the number of its boundary edges. In this chapter, we exploit this behavior in the case of regular hexagonal triangulations such as those of Fig. 5.1(a,b), which can easily be stitched together to model more complex surfaces such as the t-shirt of Fig. 5.1(c).

More specifically, the regular grid of Fig. 5.1(a) has $N_x \times N_y$ vertices, $N_e^b = 2(N_x - 1) + 2(N_y - 1)$ boundary edges, and therefore $2\,N_x + 2\,N_y - 1$ degrees of freedom, which is much smaller than the $3\,N_xN_y$ it would have without the inextensibility constraints.

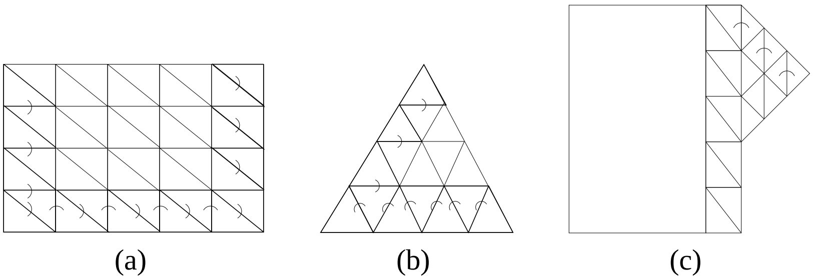

(a) (b) (c)

Figure 5.2: Specifying the 3D shape of the rectangular mesh and subdvided triangle. (a) We fix the shape of the bottom row from left to right by rotating each facet with respect to its left neighbor. For each following row, we only need to set the angle between the leftmost facet and the one below and the angle between the rightmost facet and its left neighbor. (b) The angles between the facets of the bottom row are first set from left to right. For each upper row, only the angle of the first facet need be set. (c) Attaching two patches together. Because the base of each triangular patch is attached to the body, only one single angle is required to fully specify their first row.

Furthermore, this number of dofs includes the six that correspond to a rigid motion and can be ignored for our purposes. The triangulation of Fig. 5.1(b) has N_s vertices per side and was built by recursively subdividing a single triangle. It has $N_s(N_s+1)/2$ vertices and $N_e^b = 3(N_s-1)$ boundary edges, which results in $3\,N_s$ dofs instead of $3N_s(N_s+1)/2$.

The t-shirt of Fig. 5.1(c) is modeled by combining a rectangular patch for the body part and two triangular ones for the sleeves. In this case, the number of dofs of the triangular patches is reduced because they have common edges with the rectangular patch. As a result, the total number of dofs resulting from assembling the triangular and rectangular patches is less than the sum of dofs of each patch taken separately.

5.1.2 Angle-Based Parameterization

Here we show that the shape of a wide class of inextensible meshes can be parameterized in terms of a small number N_a of *determining angles* between theirs facets. We present procedures for choosing the N_a angles so that the number of degrees of freedom of Eq. 5.4 can be written as

$$N_d = N_a + 6\ , \tag{5.5}$$

where the 6 degrees of freedom added to N_a represent the rigid motion.

5.1.2.1 Simple Triangulations

Let us first consider the $N_x \times N_y$ mesh of Fig. 5.1(a). As shown in Fig. 5.2(a), if we constrain the horizontal, vertical and diagonal edges to retain their original lengths, only the facets of the bottom row and the first and last facets of each upper row need be set

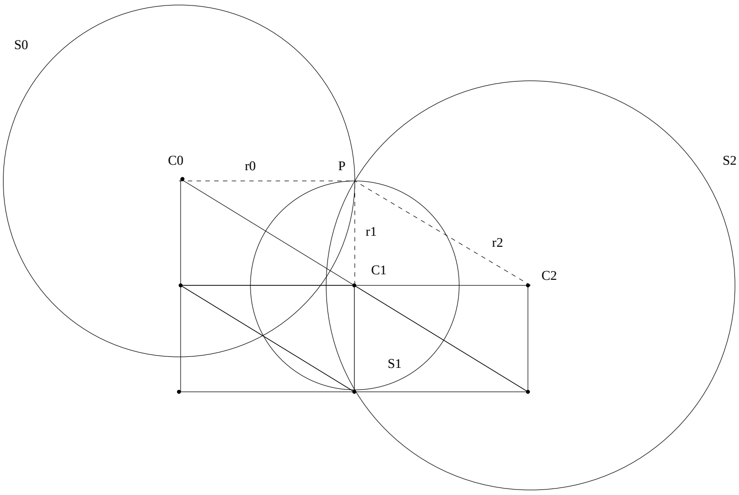


Figure 5.3: Determining the position of interior vertices by the intersection of 3 spheres. The positions of solid lines triangles have already been computed. We seek to determine the position of point $\mathbf{P}$. This can be done by computing the intersection of 3 spheres of known radii centered in $\mathbf{C}_0$, $\mathbf{C}_1$, and $\mathbf{C}_2$, respectively. This yields between two and zero solutions depending on the configuration of the other triangles.

to completely determine the shape of the grid. Each one of the remaining vertices can then be computed as the intersection of three spheres centered on previously computed vertices, as illustrated by Fig. 5.3. It can be easily checked that this requires specifying $N_a = 2(N_x - 1) + 2(N_y - 2) - 1$ determining angles and the 6 degrees of freedom that fix the position and orientation of the first facet. This corresponds to the predicted total of $N_d = 2\,N_x + 2\,N_y - 1$ dofs derived in Section 5.1.1. In other words, the chosen subset of angles gives us a model with the right number of degrees of freedom.

In the case of the subdivided triangle with N_s vertices per side of Fig. 5.1(b), we use the very similar construction depicted by Fig. 5.2(b). The total number of determining angles is $N_a = 2(N_s - 2) + (N_s - 2) = 3\,N_s - 6$. To this number, we must add the six dofs required to fix the position and orientation of the first facet in space to get the expected total of $N_d = 3\,N_s$ dofs discussed in Section 5.1.1.

5.1.2.2 Complex Triangulations

As discussed in Section 5.1.1, we modeled the t-shirt of Fig. 5.1(c) by combining a rectangular patch for the body part and two triangular ones for the sleeves. We parameterize the rectangular patch as before. As shown in Fig. 5.2(c), because the base of each triangular patch is attached to the body, only one single angle is required to fully specify their first row. The remaining rows of the triangles can then be specified as before.

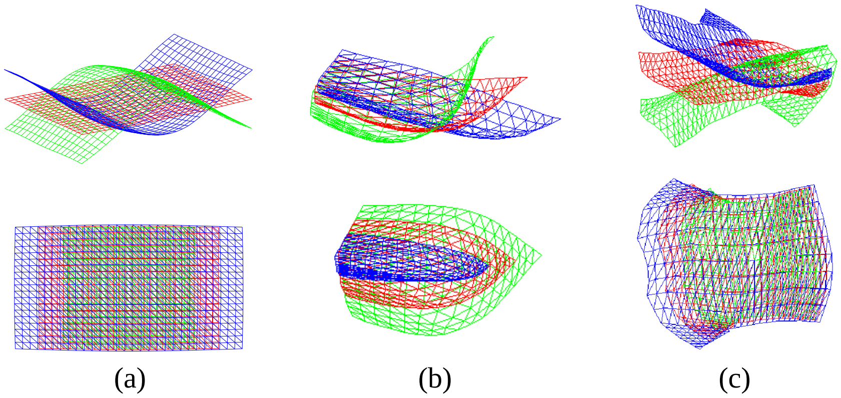

(a) (b) (c)

Figure 5.4: Deformation modes of the meshes of Fig. 5.1. In all figures, $\mathbf{y}_0$, the average mesh, is shown in red. The other two are obtained by taking a single mode weight to be non zero. A positive value of that weight yields the green mesh and a negative one the mesh shown in blue. Bending and extension modes of (a) the flat rectangular mesh, (b) the triangular spinnaker, and (c) the t-shirt.

Note that this approach is very general and could be extended to any surface without holes that can be unfolded to a planar polygon of arbitrary shape: Any polygon can be triangulated without adding any interior vertex [136]. The dual graph of such a triangulation, that is, the graph connecting the centers of neighboring facets, cannot contain any cycle because such a cycle would have to enclose at least a vertex, which would then be an interior vertex. This implies that we can build the triangulation by sequentially inserting triangles in such a way that each new one, except the first, has a single common edge with one already present. Given this order, we can represent the individual triangles as hexagonal triangulations attached to each other and parameterize them as discussed above.

5.1.3 Dimensionality Reduction

The angle-based parameterization we introduced above reduces the number of parameters required to specify the shape of an inextensible mesh. However, it is not particularly well adapted to fitting surfaces to image data for several reasons. First, it imposes an arbitrary graph structure among the vertices and specifies the coordinates of *child* vertices as a function of those of *parent* vertices, which tends to degrade the performance of optimization algorithms. Second, computing the actual shape involves solving quadratic equations representing the intersection of three spheres, which is computationally expensive. Additionally, intersecting 3 spheres may result in two solutions for some configurations or none at all for others, which makes this parameterization impractical for optimization purposes. Finally, its number of dofs still depends on the mesh resolution.

We therefore use the angle-based parameterization as an intermediate representation that lets us sample the set of possible shapes by randomly drawing the angles from a uniform distribution between two bounds. In practice, the angles were drawn in the range

$[-\pi/6, \pi/6]$, or in the range $[-\pi/9, \pi/9]$, depending on the expected flexibility of the surface at hand. The sampled shapes can then be used as training data for dimensionality reduction techniques, which will yield the true deformation model. Our representation in terms of determining angles gives us an enormous advantage over usual training data acquisition methods, since the resulting shapes are all deformed version of a single topology. Therefore there is no need of the usual painstaking 3D scan alignment and remeshing process, as, for example, was the case in [19].

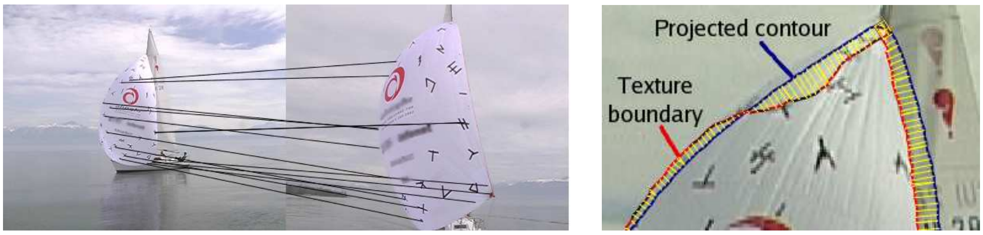
Since all the resulting deformed meshes have the same topology, we form a $3N_v$ vector for each one by concatenating the coordinates of its N_v vertices. We then apply Principal Component Analysis on these vectors, which involves the eigen-decomposition of the data covariance matrix. By retaining only the first $N_m \leq N_d << 3N_v$ principal components, we can approximate the vector of coordinates of any mesh as

$$\mathbf{y} = \mathbf{y}_0 + \sum_{k=1}^{N_m} \mathrm{x}_k \mathbf{s}_k \ , \tag{5.6}$$

where $\mathbf{y}_0$ is the vector corresponding to an undeformed mesh, the $\mathbf{s}_k$ are the principal components or modes, and the x_k are weights that specify the surface shape and act as low-dimensional latent variables. In other words, the shape of a mesh can now be expressed as a function of the vector $\mathbf{x} = [\mathrm{x}_1 \ ... \ \mathrm{x}_{N_m}]^T$. Note that, in some sense, this is similar to modal analysis where the object's behavior is described by superposing its natural strain and vibration modes [122]. However, unlike modal analysis, we do not require the kind of physical knowledge that building the appropriate stiffness matrix requires.

Fig. 5.4 depicts the influence of two of the most significant modes in the case of the meshes of Fig. 5.1. Giving weight to the first produces bending and, to the second, extension. The presence of extension modes may seem surprising since all the samples we used to learn the model are instances of the same inextensible mesh. However, given that the deformations are not linear when expressed in terms of 3D coordinates, there is no reason for the manifold of all resulting shapes to lie on a hyperplane. Intuitively, by using PCA, we consider the N_m-dimensional ellipsoid that includes this manifold without being limited to it. This produces not only extension modes but also rigid ones that we discard.

In practice, the presence of these extension modes makes the method more general: On one hand, if the surface whose deformations we seek to recover is truly inextensible, we can incorporate a term that prevents extension or shrinking into our optimization scheme. On the other hand, the presence of the extension terms lets us effectively model stretchable materials using a low-dimensional deformation model. In theory, it should be possible to remove those extension modes by replacing PCA by a non-linear dimensionality reduction technique. However, this would not help much without using a database that is much closer to the true physics. This is because a non-linear technique is very likely to force the model to stick much closer to the training data. In some sense, that would negate one of the strengths of our approach that does not require either accurate training data or precise knowledge of the physics, both of which are often hard to obtain. Furthermore, as shown in



(a) (b) (c)

Figure 5.5: Image data. (a) An image from an input sequence. (b) One of 15 images used to build a textured 3D model of the spinnaker. For our experiments, we added black scotch tape on the otherwise white parts of the sail to help our wide-baseline algorithm to find correspondences between model and input images such as those depicted by the black lines. (c) Contours detected as texture boundaries. Even though the boundary is not correct everywhere, thanks to the model and robust estimation we still recover the correct shape.

Chapter 7, learning a non-linear global model from real data proved intractable in practice due to the memory requirements and the complexity of the learning algorithm.

5.2 Shape Recovery by Tracking

We outline here our approach to using our models to take full advantage of the available image information, acquired using one or more cameras, while ignoring erroneous data. Note that this approach is defined as tracking, since it relies on image-to-image correspondences in addition to the usual model-to-image matches. However, it does not rely on any motion model as the methods described in the previous chapter.

Recall from Section 5.1.3 that the shape of the mesh is controlled by the vector $\mathbf{x}$ of weights assigned to the PCA modes. To handle the potentially moving camera, or cameras, we introduce a vector of extrinsic parameters κ for each one and define the state vector

$$\Theta = \left[\kappa_1, \ldots, \kappa_{N_p}, \mathbf{x}\right]^T \quad , \tag{5.7}$$

where N_p is the number of cameras being used. Note that this formulation can handle both one single camera and multiple cameras that may move with respect to each other.

We use the image data to write N_o observation equations of the form

$$O^i(\,\cdot\,, \Theta) = \epsilon_i \ , \ 1 \leq i \leq N_o \ , \tag{5.8}$$

where O^i is a differentiable objective function associated to a particular type of image data, and ϵ_i an error term. Here we consider the functions O^m, O^t and O^c derived from model-to-image point correspondences, image-to-image point correspondences, and contour information respectively.

Model-to-image correspondences. As shown in Fig. 5.5(a,b), when a textured 3D model of the object in its rest position is available, we use a fast wide-baseline feature matching technique [99] to compute correspondences between surface 3D locations $\mathbf{q}$ and 2D image features. We define $O^m(\mathbf{q}, \Theta)$ as the Euclidean distance in the image plane between the projection of $\mathbf{q}$ and the corresponding image feature.

Image-to-Image Correspondences. Given a couple $\mathbf{m}_i = (\mathbf{u}_i^1, \mathbf{u}_i^2)$ of corresponding points in two different images of the surface found using the same technique as before [99], we define $O^t(\mathbf{m}_i, \Theta)$ as follows: We back-project $\mathbf{u}_i^1$ to the 3D surface and reproject it into the second image. We then take $O^t(\mathbf{m}_i, \Theta)$ to be the Euclidean distance in the image plane between this reprojection and $\mathbf{u}_i^2$.

Boundary and Occluding Contours. As shown in Fig. 5.5(c), given the last known shape of the target object, we predict the location of the projection boundaries and occluding contours. We then sample these 2D contours and look for the closest edge or texture boundary in the normal direction [138]. We take O^c to be the Euclidean distance between the projection and the image edges.

As we saw in Section 5.1.3, a linear combination of principal components can result in a mesh that expands or shrinks. To model surfaces that do not stretch, we force edge lengths to remain constant by introducing a penalty term

$$E_D = \sum_{i=1}^{N_v} \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \left(\|\mathbf{v}_i - \mathbf{v}_j\| - L_{i,j} \right)^2 \quad , \tag{5.9}$$

where $\mathbf{v}_i$ is a vertex of the mesh, $\mathcal{N}(\mathbf{v}_i)$ represents the set of all its neighbors, and $L_{i,j}$ is the initial edge length. Finally, we take the global objective function E we minimize to be

$$E = \frac{1}{2} \sum_{i=0}^{N_o} w_i \rho \left(\left\| O^i(\cdot , \Theta) \right\|^2 , r \right) + w_D E_D \quad , \tag{5.10}$$

where the w_i is the weight associated to the particular type of observation i and designed so that the derivatives of all observations are of commensurate magnitude, w_D is a user-defined weight, and ρ a robust estimator whose radius of confidence r progressively decreases during the optimization. As discussed in [127], this scheme allows convergence from arbitrary starting positions. A small, or zero, w_D lets the mesh stretch or shrink. Note, that besides the term that constrains the length, we introduce no other shape regularization term.

5.3 Experimental Results

In this section, we present results obtained with our tracking algorithm. Using synthetic data, we first show that the linear model is sufficient to recover complex 3D shapes, and that

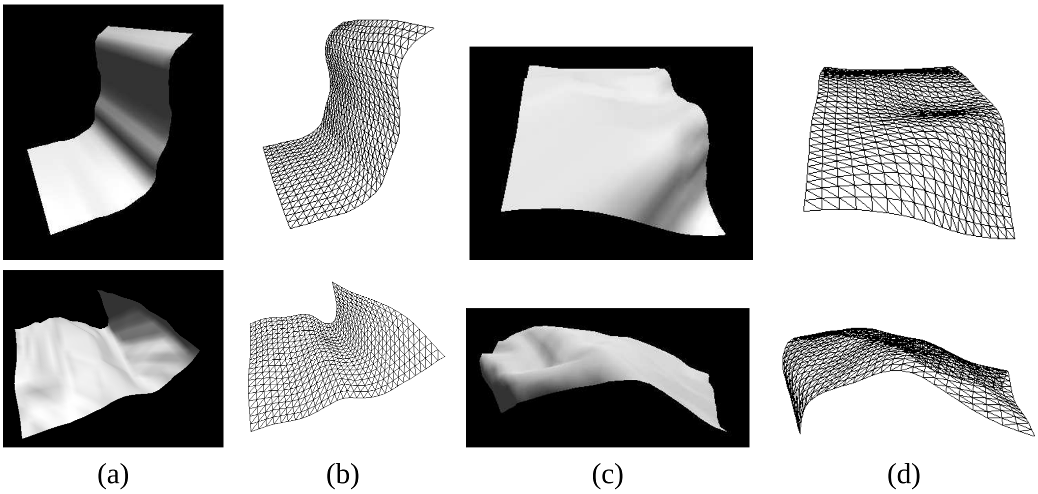


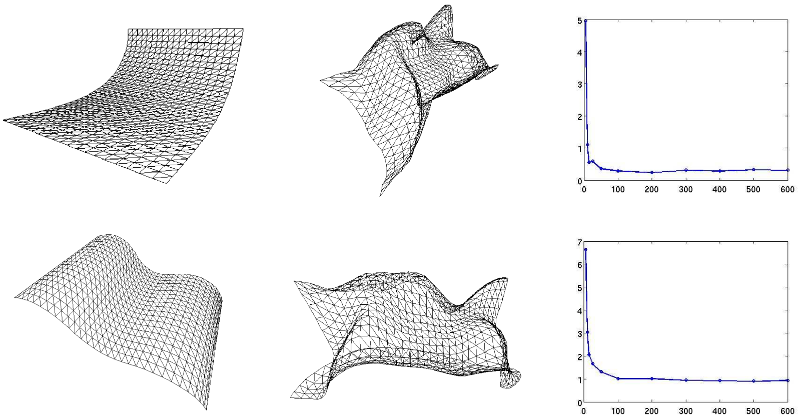
Figure 5.6: Fitting test surfaces created by varying the determining angles. Using 50 deformation modes proved sufficient to reconstruct the surfaces to a good precision. (a,c) The original shapes are shown as shaded. (b,d) The fitted ones are displayed as wireframes.

the reconstruction is insensitive to initial conditions. Finally, we present results obtained on real sequences.

5.3.1 Synthetic Data

Recall from Section 5.1.2 that we created the deformed mesh samples by varying a number of determining angles between the mesh facets. Arguably, using a much smaller number of principal components could fail to cover all possible such deformations and result in principal components unable to describe some configurations. To disprove this, we generated a number of test meshes such as the ones of Fig. 5.6(a,c) by randomizing the determining the angles in a similar fashion. We then reconstructed them by minimizing the 3D vertex-to-vertex distance with respect to the PCA coefficients $\mathbf{x}$. As can be shown in Fig, 5.6(b,d), using $N_m = 50$ principal components was sufficient to accurately fit the resulting shapes.

To demonstrate that our optimization process is well-posed and insensitive to initial conditions, we ran our system on synthetic data. We created randomly deformed versions of the rectangular mesh such as the ones shown in the first column of Fig. 5.7 and used them to produce large numbers of synthetic model-to-image correspondences. For each test-run, we started from a different random initialization such as the ones of the second column of Fig. 5.7 and minimized the objective function of Eq. 5.10 using random subsets of the correspondences. In the third column of the figure, we plot the median of the mean distances between the vertices of the recovered mesh to the correct one as a function of the number of correspondences that were used. More precisely, given a number n between 5 and 600, we picked four different subsets of n correspondences and ran the algorithm with 100 dif-



(a) (b) (c)

Figure 5.7: Convergence using synthetic data. (a) Projections of the synthetic surfaces used as input for the optimization process. (b) Examples of initializations. (c) Median of the mean distances between the vertices of the recovered mesh and the synthetic surface as a function of the number of correspondences that were used. The measures are given as a percentage of the longest side of the initial rectangle. We did not draw error bars because, as soon as we used more than 50 matches, the first and third quartile of the mean distances are indistinguishable from the median.

ferent initial shapes for each. As soon as enough correspondences are used, the algorithm consistently converges towards the correct solution.

5.3.2 Real Images

Here we demonstrate the capabilities of our method to recover the possibly large deformations of different kinds of objects that cover a wide range of physical properties. The images have been acquired using ordinary camcorders. First, we present pure tracking results obtained by using image-to-image correspondences and silhouettes only, and then we show our results of combined tracking and detection by adding model-to-image matches.

5.3.2.1 Tracking Results

We first applied our method to tracking deformable surfaces in monocular sequences using frame-to-frame matches and silhouettes. This assumes, as in the previous chapter, that the shape in the first frame in known. For these results, we further assumed that the camera is fixed, and thus the state vector Θ only contains the shape parameters $\mathbf{x}$. Simply keeping the number of principal components we use low is enough to enforce smoothness without having to explicitly add a regularization term. However, in some cases, we had to fix some coordinates of the meshes to avoid ambiguities due to the chosen viewpoints.

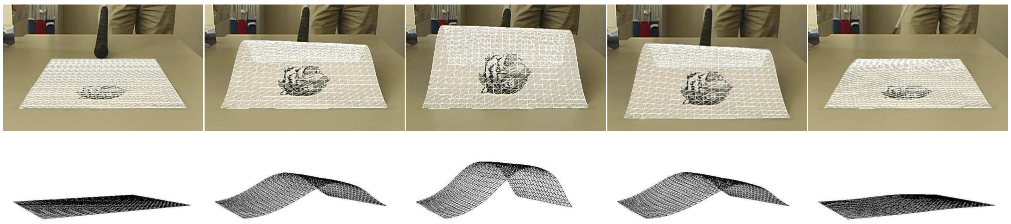


Figure 5.8: Deforming a sheet of paper. Top row: Deformed mesh projected on the original sequence as a wireframe. Bottom row: Deformed mesh shown as a wireframe model seen from a different viewpoint. Note that even the back deforms correctly.

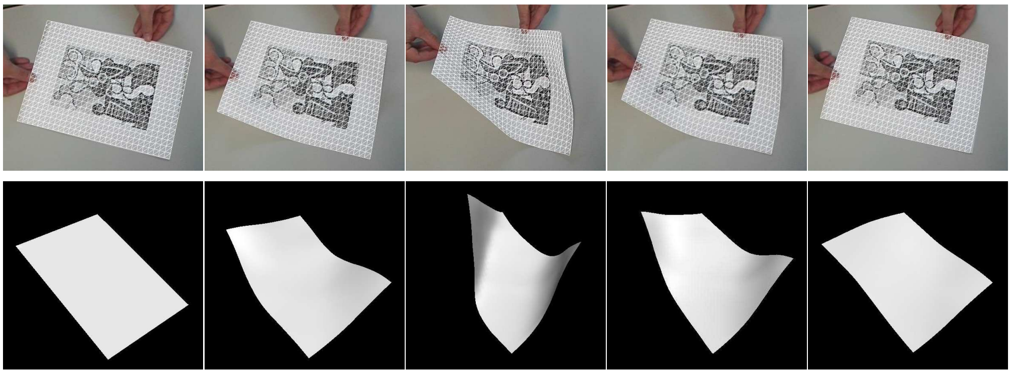


Figure 5.9: Another deforming sheet. Top row: Projected wireframe. Bottom row: Deformed mesh shaded and seen from a different viewpoint.

Fig. 5.8 depicts the tracking of a piece of paper starting from an undeformed position. Even though there is texture at only one place on the paper, the whole model deforms correctly. This includes the back of the sheet that is not actually seen in the video. Another deforming sheet of paper is shown in Fig.5.9. The chosen viewpoint makes it difficult to clearly see the deformation in the first frames. Our algorithm nevertheless retrieves the precise 3D shape throughout the whole sequence. In both cases, we used 30 principal components.

Fig. 5.10 shows the behavior of our algorithm when applied to a cloth-like material that is more flexible and required the use of 45 principal components instead of the 30 used before. The deformation is mostly perpendicular to the image plane, which again makes it challenging to track. As can be seen in the figure, the reprojected shape closely matches the object in the images, except occasionally near the corners. This can be attributed to the fact that, because the fabric is very textured, our approach to detecting edges can become confused and should be replaced by a more sophisticated one.

As mentioned before, some of the PCA modes account for extension and stretching. Therefore, we used an inflating and deflating balloon to test our algorithm's behavior when

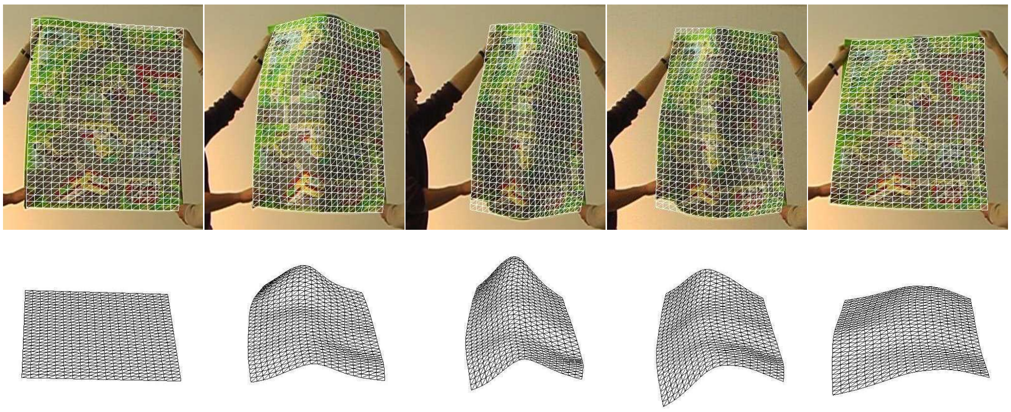


Figure 5.10: Deforming fabric. The results are displayed in the same manner as in Fig 5.8. Since the fabric is highly textured, borders of the mesh are sometimes mismatched with texture edges, which results in small misalignments.

the surface can globally stretch or shrink. In all the balloon examples presented here, the initial mesh shapes were obtained by scanning the balloons before starting inflation or deflation and fitting our mesh models to the scans.

All the results shown above involved the use of the penalty term E_D of Eq. 5.9 to force the mesh edges to retain their original lengths. In Fig. 5.11, we allow the mesh to stretch by setting the weight of this E_D term to zero. The last row of the figure displays an augmented version of the mesh. Indeed, having the 3D mesh lets us re-texture it, and project it onto the original image, thus creating a new virtual balloon. Since we are not tracking the whole balloon, but only its textured part, we only use correspondences and ignore edges. The mesh then expands along with the balloon, which is made possible by principal components such as the one depicted by the bottom row of Fig. 5.4. As shown in Fig. 5.12, the opposite behavior is observed when the balloon deflates.

Finally, Fig. 5.13 depicts the results obtained on the same sequence as in Fig. 5.11 when penalizing the edge length variation. Because the surface cannot stretch, it ends up covering a smaller fraction of the balloon and gets flatter as the balloon inflates. This also is a correct representation if we do not assume that the mesh is attached to a precise area of the surface, but rather models the behavior of a fixed size part, which is valid since we use image-to-image correspondences only. Our parameterization does not force any vertex to correspond to one particular point on the object and the mesh can slide along the object surface. Fig. 5.14 shows the superposed meshes in the first and last frame of the sequence for the case where extension is not penalized, and for the case where it is. It can indeed be noted that the final shapes are rather different, but can both be considered as correct depending on the expected behavior of the model.

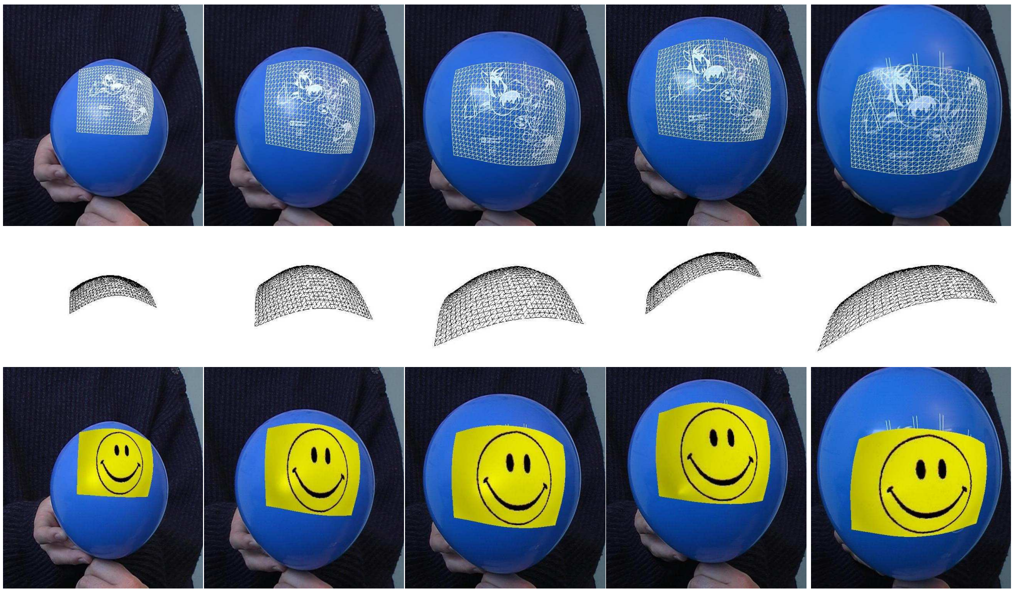


Figure 5.11: Tracking an inflating balloon with an extensible mesh. Note that the mesh keeps on covering the same portion of the balloon. In the last row, we re-textured the resulting mesh, and reprojected it into the images.

5.3.2.2 Tracking and Detection

We now present results obtained by optimizing the full criterion of Eq. 5.10 using model-to-image correspondences in addition to correspondences with the previous image, and optionally silhouette information. This runs at between 0.3 and 2.5 frames/second when using 40 modes and 1024x768 images, the faster rate being obtained when not using silhouettes. Recall from Section 5.2 that we use an optimization schedule that lets us start from arbitrary positions, which means that this does not require any manual intervention at run-time. As shown in the figures of this section, the resulting shapes are accurate enough for correct reprojection. However, enforcing temporal consistency only over image pairs might leave a residual 3D jittering motion across frames when creating videos. Therefore, this jitter can be eliminated by reoptimizing our criterion using the same observations as before, but over larger sets of 8 overlapping frames, and enforcing temporal coherence by penalizing the second derivatives of all parameters.

We represent both the sheet of paper and the elastic surface, which was cut out of an inflatable balloon, of Fig. 5.16 and 5.18 as a 30×20 rectangular grid. We model the spinnaker using a 153-vertex triangle. The t-shirt mesh is made of 2 sleeves that are 45-vertex equilateral triangle attached to a 9×25 rectangular grid. We used 45 PCA modes to track the sheet of paper, 10 for the balloon, 40 for the spinnaker and 50 for the t-shirt.

The rest shape of the spinnaker is not planar and NorthSails$^{\mathrm{TM}}$ gave us the CAD model that was used to design it, thus allowing us to fit a triangular mesh to it. This gave us

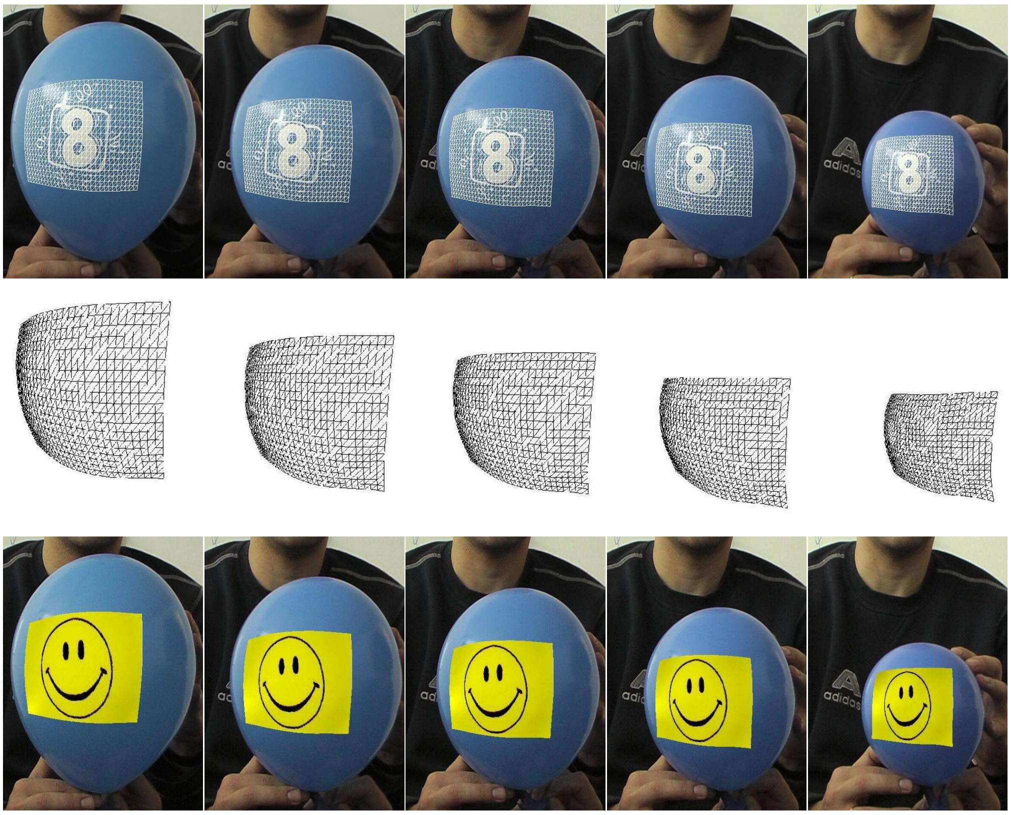


Figure 5.12: The inverse behavior as in Fig. 5.11 can be observed when the balloon deflates.

the initial shape from which we computed the deformation modes. To create the textured model needed for automated run-time operation, we developed a software tool that allows us to manually supply a few image-to-model correspondences in images such as those of Fig. 5.15, which do *not* belong to the test video. By feeding these correspondences along with automatically detected silhouettes into the optimization framework of Section 5.2, we recover the spinnaker's shape into the images and, thus, a textured model. In the specific case depicted by Fig. 5.15, we only supplied 10 correspondences per image, which did not take long to do. In short, our deformation modes not only lead to robust and automated run-time operation but can also be used to limit the required amount of manual intervention during model building.

These experiments display several strengths of our method. First, for the deforming sheet of paper of Fig. 5.16 and spinnaker of Fig. 5.17, our system proved robust enough to process sequences of more than 1500 frames acquired both indoors and outdoors without getting lost or drifting. When the image data was too weak, 3D shape recovery became temporarily less accurate but the system soon recovered.

The t-shirt example of Fig. 5.16 shows that even though the resulting PCA modes repre-

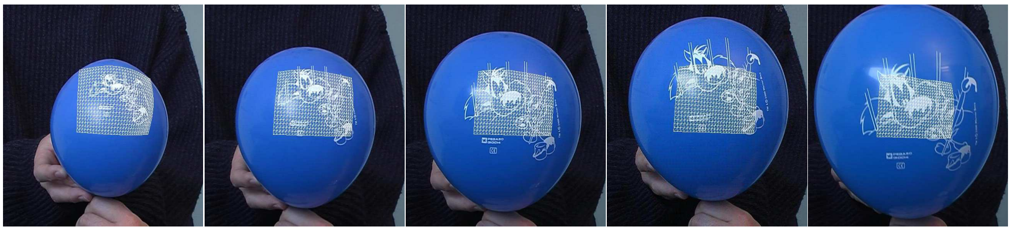


Figure 5.13: Tracking an inflating balloon with an inextensible mesh. The portion of the balloon covered by the mesh becomes smaller and flatter as the balloon expands. This also is a correct solution if we do not assume that the mesh represents a particular portion of the object, but rather models the behavior of a fixed size part, which is valid when relying on image-to-image matches only.

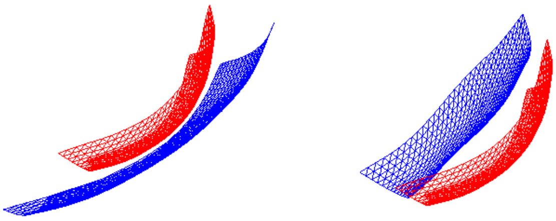


Figure 5.14: Superposition of the initial mesh in red and the final one in blue. Left: When extension is not penalized, the mesh increases as the balloon inflates. Right: When we enforce the edges to remain of constant length, the surface remains of same area, but becomes flatter.

sent global deformations, we can still track very local ones, such as only one moving sleeve, by superposing these modes. Finally, in the case of the stretching surface of Fig. 5.18, we can see that not only global extension can be modeled, but also anisotropic stretching. This, again, is due to the fact that local deformations can be accurately described by appropriately superposing global modes.

All these results were generated using a single video sequence per object except in the spinnaker case where we used either one or two cameras. In this case, the two cameras were hand-held by two people on a chase-boat so that they move with respect to each other in an unpredictable fashion and do not form a stereo-rig in the usual sense of the term. Our framework is powerful enough to handle this case and to take full advantage of all the available information, even in such non-standard conditions. As shown in Fig. 5.17, once reprojected on the images, the results are almost indistinguishable. Of course, because a single camera cannot see both sides of a curvy object, the quality of the 3D results is bound to be better when using two cameras looking from very different angles so as to see different parts of the object. However, the superposition of both 3D results in Fig. 5.17(e) shows that, in this case, the model approximates the hidden part well. This behavior is

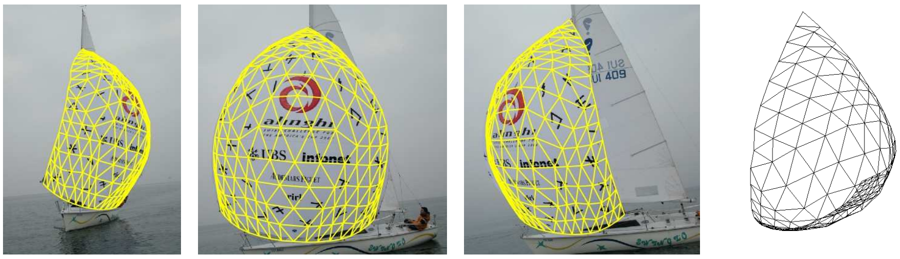


Figure 5.15: 3D model of the spinnaker overlaid on the three images used to compute its reference shape and texture. In each image, we specified 10 correspondences with a CAD model of the spinnaker and used them, along with automatically detected silhouettes, to deform it. We assume that the spinnaker did not deform in these images because they were taken in quick succession by a chase-boat.

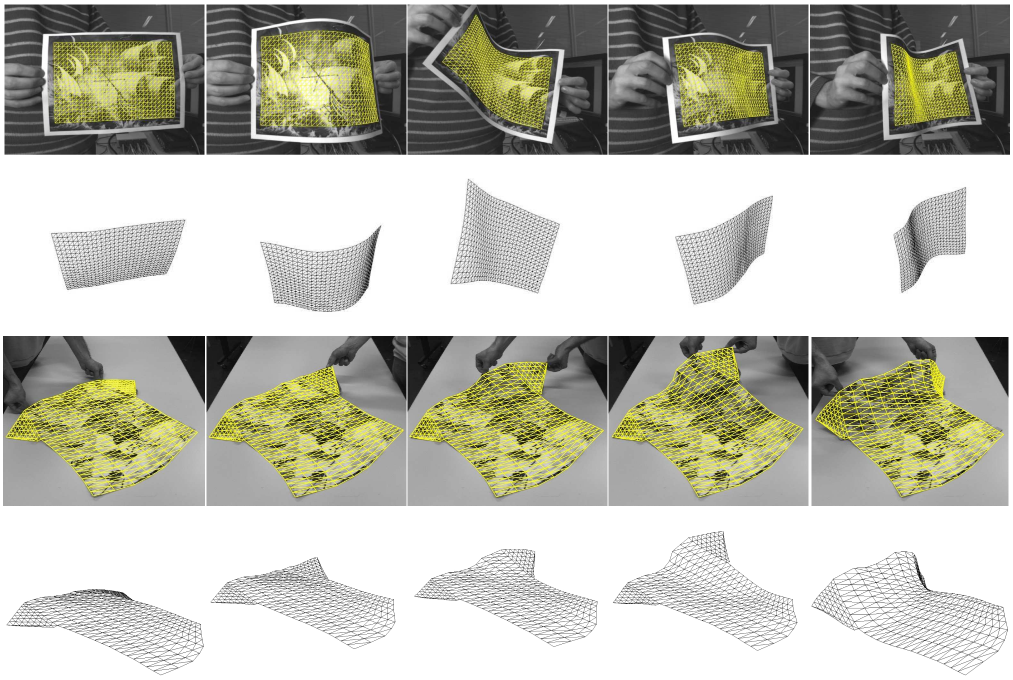


Figure 5.16: Tracking a deforming sheet of paper and a t-shirt. In both cases, we show the deformed 3D mesh overlaid on the original images in the top row and then seen from a different viewpoint in the bottom row.

consistent over the whole video sequence.

Finally, to estimate the accuracy of our reconstructions, we acquired three videos of another sheet of paper using three calibrated and synchronized cameras. We used one to monocularly reconstruct the deforming shape using our method. We used the other two

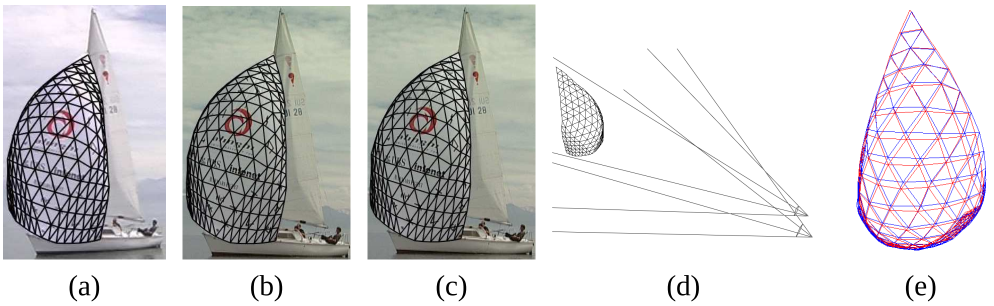


Figure 5.17: Tracking a spinnaker with either one or two cameras. (a,b) Two synchronized images from independently moving cameras, with recovered spinnaker reprojection. (c) Tracking using only one camera. Note that, once reprojected on the images, the results are almost indistinguishable. (d) 3D results with two cameras. Both camera positions are also retrieved. (e) Superposed 3D shapes retrieved using either one (red) or two (blue) cameras. Note that both shapes are very similar, which indicates that the deformation model provides a good approximation when data is missing.

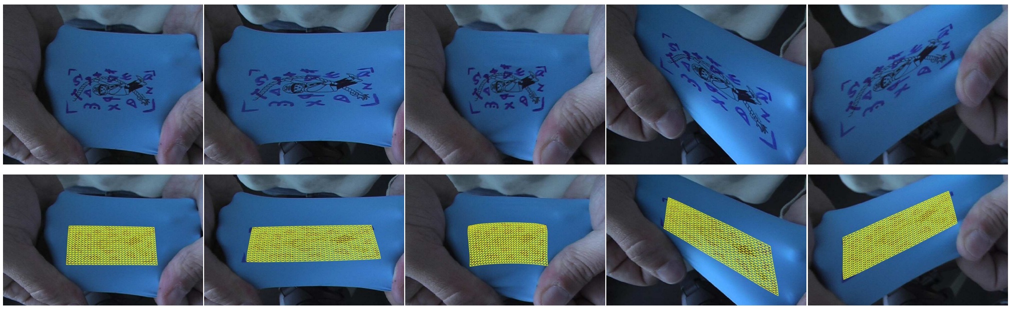
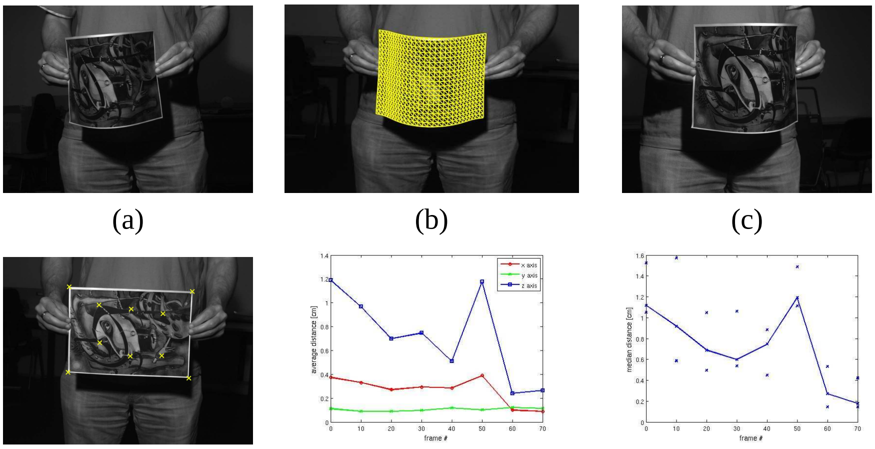


Figure 5.18: Tracking an extensible surface undergoing anisotropic deformations. In the top row, we show the original images and, in the bottom row, we overlay the recovered 3D grid that stretches appropriately.

to triangulate the x-, y-, and z-coordinates of 10 selected points by manually establishing correspondences every 10 frames of the 70 frames-long sequences. These points—the four corners of the sheet plus six additional ones spread over its surface—are depicted by Fig. 5.19(d) and were chosen to be representative of the whole surface. As shown in Fig. 5.19(e), the largest average errors occur in the z-direction, which was to be expected since it is close to the viewing direction. In Euclidean terms, this corresponds to the median errors of Fig. 5.19(f), which are in the order of 1cm. This is quite small considering that this was achieved using a single camera that was approximately 1.5 meter away from the 29.7cm $\times 21.0$cm rectangular sheet of paper.



(a) (b) (c)

(d) (e) (f)

Figure 5.19: Evaluating the accuracy of our approach. (a,b,c): Images from videos acquired using three synchronized and calibrated cameras. Image (b) belongs to the video we used to monocularly reconstruct the 3D shape using our method and, then, re-projecting it into the image. (d) We triangulated the 3D coordinates of the 10 keypoints shown as crosses by manually establishing correspondences in images (a) and (c). (e) We repeated this operation every 10 frames and plot the average differences between the x-, y-, and z-coordinates of those manually computed and those derived from our automated and monocular reconstruction. (f) We also computed the Euclidean distances between the monocular reconstructions and the manually computed points, and plot their medians together with values at 25% and 75%.

5.4 Closed-Form Solution to Non-Rigid 3D Registration

In Section 5.2, we have presented an optimization approach to recovering the shape of a non-rigid surface from a video sequence. In this section, we will now show that, when using model-to-image correspondences only, this can also be done in closed-form. The major advantage of this is that we can now reconstruct a deformable surface from individual images and a reference configuration. Furthermore, this also completely prevents the results from drifting troughout a video sequence.

As in Section 3.2.1, we first formulate 3D reconstruction as the solution to a linear system. We then show that the depths ambiguities can be overcome by solving a system of quadratic equations accounting for constant edges lengths, which can be done in closed-form using linearization techniques. Since modeling the deformations in terms of the vertices coordinates involves too many parameters for the problem to be tractable, we then show how our global models can be introduced in this framework.

5.4.1 Linear Formulation

Let us consider the 3D registration of a mesh to an input image, given a reference configuration. As in Chapter 3, we assume that we are given a set of N_c 3D-to-2D correspondences between the surface and the image. Each correspondence relates a 3D point on the mesh, expressed in terms of its barycentric coordinates with respect to the vertices of the facet to which it belongs, and a 2D feature in the image.

Additionally, we assume the camera to be calibrated and, therefore, that its matrix of intrinsic parameters $\mathbf{A}$ is known. To simplify our notations without loss of generality, we express the vertex coordinates in the camera referential.

Let us first set aside our linear deformation model and consider the reconstruction of the 3D coordinates of the N_v mesh vertices $\mathbf{v}_i$, $1 \leq i \leq N_v$. Recall from Section 3.2.1 that, in this framework, the correspondence problem can be formulated as the solution to the linear system

$$\mathbf{M}\mathbf{y} = \mathbf{0} \ , \tag{5.11}$$

where $\mathbf{M}$ is the $2N_c \times 3N_v$ matrix of Eq. 3.6, and $\mathbf{y} = \left[\mathbf{v}_1^T \ ... \ \mathbf{v}_{N_v}^T\right]^T$.

As detailed in Section 3.2.1, although solving this system yields a surface that reprojects correctly on the image, there is no guarantee that its 3D shape corresponds to reality. This stems from the fact that, for all practical purposes, $\mathbf{M}$ is rank deficient. More specifically, even when there are many correspondences, one third, i.e. N_v, of the eigenvalues of $\mathbf{M}^T\mathbf{M}$ are very close to zero, as illustrated by Fig. 5.20(c). As a result, even small amounts of noise produce large instabilities in the recovered shape.

This suggests that additional constraints have to be added to guarantee a unique and stable solution. Following what was proposed in the previous part of this chapter, we will argue that imposing inextensibility of the surface yields a closed-form solution to the problem.

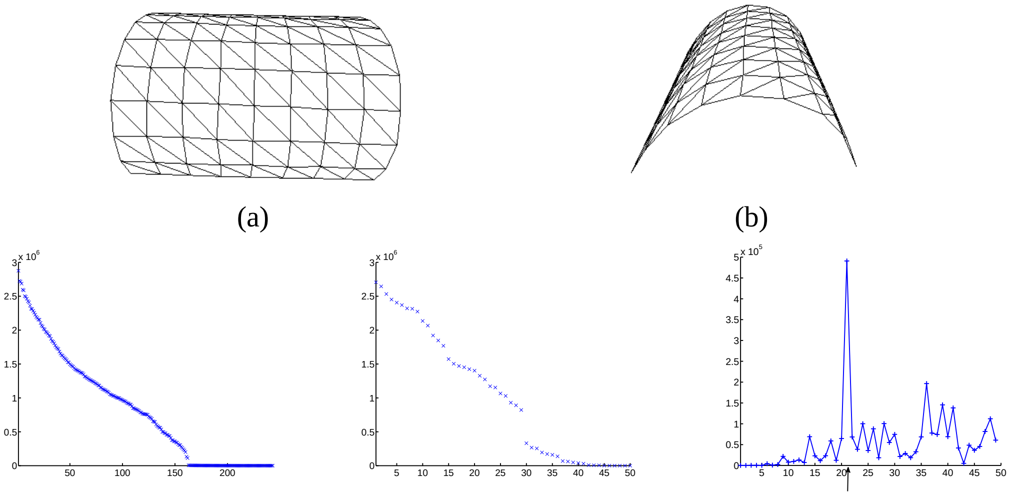
5.4.2 Inextensibility Constraints

Following the idea introduced in [111] in the context of rigid object pose recovery, we write the solution of the linear system of Eq. 5.11 as a weighted sum of the eigenvectors $\mathbf{l}_i$, $1 \leq i \leq N_v$ of $\mathbf{M}^T\mathbf{M}$, which are those associated with the eigenvalues that are almost zero. Therefore we write

$$\mathbf{y} = \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i \ , \tag{5.12}$$

since any such linear combination of $\mathbf{l}_i$ is in the kernel of $\mathbf{M}^T\mathbf{M}$ and produces a mesh that projects correctly on the image. Our problem now becomes finding appropriate values for the β_i, which are the new unknowns.

We are now in a position to exploit the inextensibility of the surface by choosing the β_i so that lengths of the N_e edges are preserved. Such β_i can be expressed as the solution of



(a) (b)

(c) (d) (e)

Figure 5.20: (a,b) Original and side views of a surface used to generate a synthetic sequence. The 3D shape was reconstructed by an optical motion capture system. (c,d) Eigenvalues of the linear system written from correspondences randomly established for the synthetic shape of (a). (c) The system was written in terms of 243 vertex coordinates. One third of the eigenvalues are close to zero. (d) The system was written in terms of 50 PCA modes. There are still a number of near zero eigenvalues. (e) First derivative of the curve (d) (in reversed x-direction). We take the maximum value of N_l to be the one with maximum derivative, which corresponds to the jump in (d).

a set of quadratic equations of the form

$$\left\| \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^j - \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^k \right\|^2 = \left\| \mathbf{v}_j^{ref} - \mathbf{v}_k^{ref} \right\|^2 , \tag{5.13}$$

where $\mathbf{l}_i^j$ is the 3×1 sub-vector of $\mathbf{l}_i$ corresponding to the coordinates of vertex $\mathbf{v}_j$, and $\mathbf{v}_j^{ref}$ and $\mathbf{v}_k^{ref}$ are two neighboring vertices in the reference configuration.

Typical closed-form approaches to solving systems of quadratic equations involve linearizing the system and introducing new unknowns for the quadratic terms. This results in a system of the form

$$\mathbf{Db} = \mathbf{d} , \tag{5.14}$$

where $\mathbf{b} = [\beta_1\beta_1, \cdots, \beta_1\beta_{N_v}, \beta_2\beta_2, \cdots, \beta_2\beta_{N_v}, \cdots, \beta_{N_v}\beta_{N_v}]^T$ is the vector of quadratic terms of size $N_v(N_v+1)/2$. $\mathbf{D}$ is a $N_e \times N_v(N_v+1)/2$ matrix built from the known $\mathbf{l}_i$, and $\mathbf{d}$ is the $N_e \times 1$ vector of edge lengths in the reference configuration. Unfortunately, since, in hexagonal meshes, the number of edges grows as $3N_v$, the number of quadratic unknown terms in the linearized system quickly becomes larger than the number of equations.

Here, we solve this problem by using Extended Linearization [40], a simple and powerful approach to creating new equations in a linearized system. It has been shown to perform better than Groebner bases with a large number of parameters, and can be considered as a generalization of relinearization which introduces new equations that link the quadratic terms together to make them coherent. The idea is to multiply the original set of equations by the monomials, and linearize the resulting system. In our particular case, we can, for example, multiply the existing quadratic equations by each of the linear terms, thus creating new equations of the form

$$\begin{aligned}
\beta_1 \left(\left\| \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^j - \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^k \right\|^2 \right) &= \beta_1 \left(\left\| \mathbf{v}_j^{ref} - \mathbf{v}_k^{ref} \right\|^2 \right) \ , \\
&\vdots \\
\beta_{N_v} \left(\left\| \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^j - \sum_{i=1}^{N_v} \beta_i \mathbf{l}_i^k \right\|^2 \right) &= \beta_{N_v} \left(\left\| \mathbf{v}_j^{ref} - \mathbf{v}_k^{ref} \right\|^2 \right) \ .
\end{aligned}$$

Let $\mathbf{b}^c = [\beta_1\beta_1\beta_1, \cdots, \beta_1\beta_1\beta_{n_v}, \beta_1\beta_2\beta_2, \cdots, \beta_1\beta_2\beta_{n_v}, \beta_2\beta_2\beta_2, \cdots, \beta_{n_v}\beta_{n_v}\beta_{n_v}]^T$, and $\mathbf{b}^l = [\beta_1, \cdots, \beta_{n_v}]^T$. The resulting system can be written as

$$\left[\ \mathbf{D}_{XL}^l \ \middle| \ \mathbf{D}_{XL}^q \ \middle| \ \mathbf{D}_{XL}^c \ \right] \begin{bmatrix} \mathbf{b}^l \\ \mathbf{b} \\ \mathbf{b}^c \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{0} \\ \vdots \end{bmatrix} \ , \tag{5.15}$$

with

$$\mathbf{D}_{XL}^l = \begin{bmatrix} 0 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ -\mathbf{d}_1 & 0 & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \ , \tag{5.16}$$

$$\mathbf{D}_{XL}^q = \begin{bmatrix} \mathbf{D}_1^{1,1} & \cdots & \mathbf{D}_1^{1,N_v} & \mathbf{D}_1^{2,2} & \cdots & \mathbf{D}_1^{N_v,N_v} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \ , \text{ and} \tag{5.17}$$

$$\mathbf{D}_{XL}^c = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{D}_1^{1,1} & \cdots & \mathbf{D}_1^{N_v,N_v} & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \ , \tag{5.18}$$

and where we only explicitly show the first line of the original system of Eq. 5.14 and its product with β_1. $\mathbf{D}_1^{i,j}$ stands for the coefficient on the first line of $\mathbf{D}$ corresponding to the product $\beta_i\beta_j$.

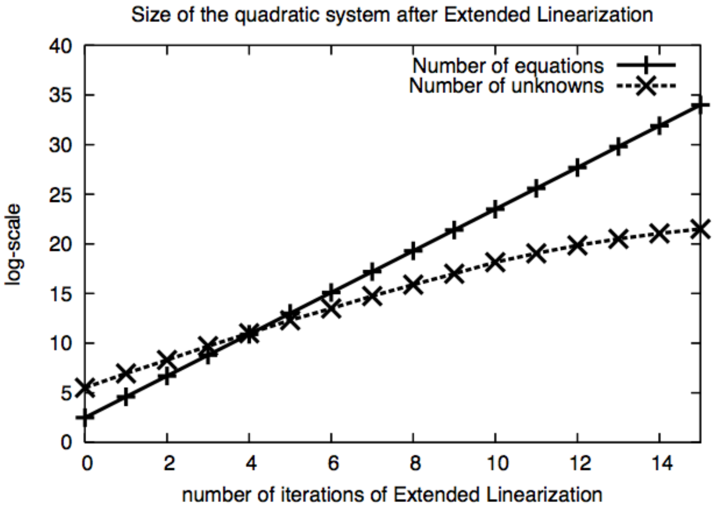


Figure 5.21: Lin-Log plot of the number of equations and number of unknowns as a function of the number of Extended Linarization iterations for the case of a 10×10 square mesh. Note that after 4 iterations, the number of equations exceeds the number of variables, making the system, in theory, solvable. However, the size of the system is of the order 10^{12}, which makes its solution intractable in practice.

Unfortunately, we can show that, when dealing with relatively large meshes, multiplying the inextensibility equations by all the β_i is not practical to solve the system of Eq. 5.14. Given a hexagonal mesh for which $N_e \propto 3N_v$, the number of equations after Extended Linearization is $N_{eq} = (N_v + 1)N_e \propto 3N_v^2 + 3N_v$. Since, the new equations also give rise to new unknowns, the linear and cubic terms, the total number of variables becomes $N_u = N_v + N_v(N_v + 1)/2 + N_v(N_v + 1)(N_v + 2)/6$. For the system to be solvable, we need

$$\begin{array}{rccc} & N_{eq} & \geq & N_u \\ \Leftrightarrow & 3N_v^2 + 3N_v & \geq & N_v + \frac{N_v(N_v+1)}{2} + \frac{N_v(N_v+1)(N_v+2)}{6} \\ \Leftrightarrow & 0 & \geq & N_v(N_v^2 - 12N_v - 7)\ . \end{array}$$

N_v being greater than zero, this implies $(N_v^2 - 12N_v - 7) \leq 0$, which is only true for $N_v \leq 12$. In practice, we rarely face cases where 12 vertices are enough to model the deformations of a surface.

Of course, Extended Linearization can be applied iteratively by re-multiplying the new equations by the linear terms. Let us assume that, at each iteration, we only consider the newly created equations, but not the original set, and that we start after a first Extended Linearization step, to avoid the special case of constant right-handside values $\mathbf{d}$. If we separate $N_u^{(0)} = N_{u,1}^{(0)} + N_{u,3}^{(0)}$ into the numbers of linear terms and cubic terms, respectively, we can then write the following recursive procedure:

$$\begin{aligned} N_{eq}^{(i+1)} &= N_v N_{eq}^{(i)} \,, \\ N_{u,1}^{(i+1)} &= N_{u,1}^{(i)} \binom{N_v + i}{i+1} \,, \\ N_{u,3}^{(i+1)} &= N_{u,3}^{(i)} \binom{N_v + i + 2}{i+3} \,, \end{aligned}$$

where a superscript (i) indicates the iteration. The number of equations therefore grows as $O(N_v^{i+1})$, whereas the number of unknowns is $O(N_v^{i+3}/(i+3)!)$. As shown in Fig. 5.21, for the case of a 10×10 vertices square mesh, after a few iterations N_{eq} indeed becomes larger than N_u. However, the size of the system is then of the order 10^{12}, which makes it impossible to solve in practice.

In other words, Extended Linearization cannot deal with a problem as large as ours and we are not aware of any other closed-form approach to solving systems of quadratic equations that could. Fortunately, we can address this issue with the help of our linear deformation model.

5.4.3 Linear Deformation Model

As discussed above, to solve the set of quadratic equations that express edge length preservation, we need to reduce its size to the point where Extended Linearization becomes a viable option. Furthermore, we need to do this in such a way that the solution of the correspondence problem can still be expressed as the solution of a system of linear equations, as discussed in Section 5.4.1.

To this end, we make use of the linear deformation model described in Section 5.1.3 that models the plausible deformations of the mesh as a linear combination of N_m deformation modes. We re-write Eq. 5.6 in matrix form as

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{S}\mathbf{x} \,, \tag{5.19}$$

where $\mathbf{S}$ is the matrix whose columns contain the deformation modes and $\mathbf{x}$ is the vector of their associated weights.

In this formulation, recovering the shape amounts to computing the weights $\mathbf{x}$. Since the shape must satisfy Eq. 5.11, $\mathbf{x}$ must then satisfy

$$\mathbf{M}(\mathbf{y}_0 + \mathbf{S}\mathbf{x}) = \mathbf{0} \,. \tag{5.20}$$

When solving this system, to ensure that the recovered weights do not generate shapes exceedingly far from our training data, we introduce a regularization term by penalizing x_i with the inverse of the corresponding eigenvalue λ_i of the data covariance matrix. This follows the probabilistic interpretation of PCA [153]. We therefore solve

$$\begin{bmatrix} \mathbf{MS} & \mathbf{M}\mathbf{y}_0 \\ w_r \Lambda^{-\frac{1}{2}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{0} \,, \tag{5.21}$$

where Λ is an $N_m \times N_m$ diagonal matrix whose elements are the λ_i and w_r is a regularization weight that only depends on the maximum λ_i, and whose precise value has only little influence on the results.

As shown in Fig. 5.20(d), we have considerably reduced the number of near-zero eigenvalues. The system of Eq. 5.21 is therefore better conditioned than the one of Eq. 5.11, but still does not yield a well-posed problem that would have a unique solution. This is attributable to the fact that, because the solution is expressed as a sum of deformation modes, inextensibility constraints, which are non linear, are not enforced.

Nonetheless, we can follow the same procedure as in Section 5.4.2. We write the solution of the linear system of Eq. 5.21 as a weighted sum of the eigenvectors $\tilde{\mathbf{l}}_i$, $1 \leq i \leq N_l \ll N_m$ associated with the smallest eigenvalues of its matrix, and find the weights $\tilde{\beta}_i$ as the solution of the linearized system of quadratic equations

$$\tilde{\mathbf{D}}\tilde{\mathbf{b}} = \tilde{\mathbf{d}} \ , \tag{5.22}$$

where $\tilde{\mathbf{b}} = [\tilde{\beta}_1, \cdots, \tilde{\beta}_{N_l}, \tilde{\beta}_1\tilde{\beta}_1, \cdots, \tilde{\beta}_1\tilde{\beta}_{N_l}, \tilde{\beta}_2\tilde{\beta}_2, \cdots, \tilde{\beta}_2\tilde{\beta}_{N_l}, \cdots, \tilde{\beta}_{N_l}\tilde{\beta}_{N_l}]^T$ now also contains the linear terms arising in the quadratic equations from the mean shape $\mathbf{y}_0$. Furthermore, the system also encodes the additionnal linear equation that constrains the $\tilde{\beta}_i\tilde{\mathbf{l}}_{i,N_m+1}$ to sum up to 1, where $\tilde{\mathbf{l}}_{i,N_m+1}$ is the last element of $\tilde{\mathbf{l}}_i$.

Since in practice $N_l \ll N_m \ll N_v$, the system is now much smaller. Therefore a single iteration of Extended Linearization is sufficient to constrain its solution while keeping it tractable, even for relatively large numbers of modes—in practice up to 60—thus allowing complex deformations.

In this formulation, the number N_l of eigenvectors strongly depends on the number N_m of modes used for the recovery. This is in contrast with [111] that only uses at most 4 of them, since they deal with rigid objects. However, as shown in Fig. 5.20(e), we can easily set the maximum number $\hat{N}_l$ of eigenvectors to use by picking the number corresponding to the maximum first derivative of the ordered eigenvalues curve. We then simply test for all $N_l \leq \hat{N}_l$ and pick the optimal value as the one that, for a small enough reprojection error, gives the smallest mean edge length variation. In practice, $\hat{N}_l$ was typically about 25 when using 60 deformation modes, while the value of N_l that generates the best results was around 10.

5.5 Experimental Results

In this section, we present the reconstructions obtained in closed-form from individual images and a reference image. Even though some of the results were computed from video sequences, nothing relates the frames of the sequences together, and no initialization is required. We first present results on synthetic data to quantitatively evaluate our reconstruction accuracy, and then show results on real images.

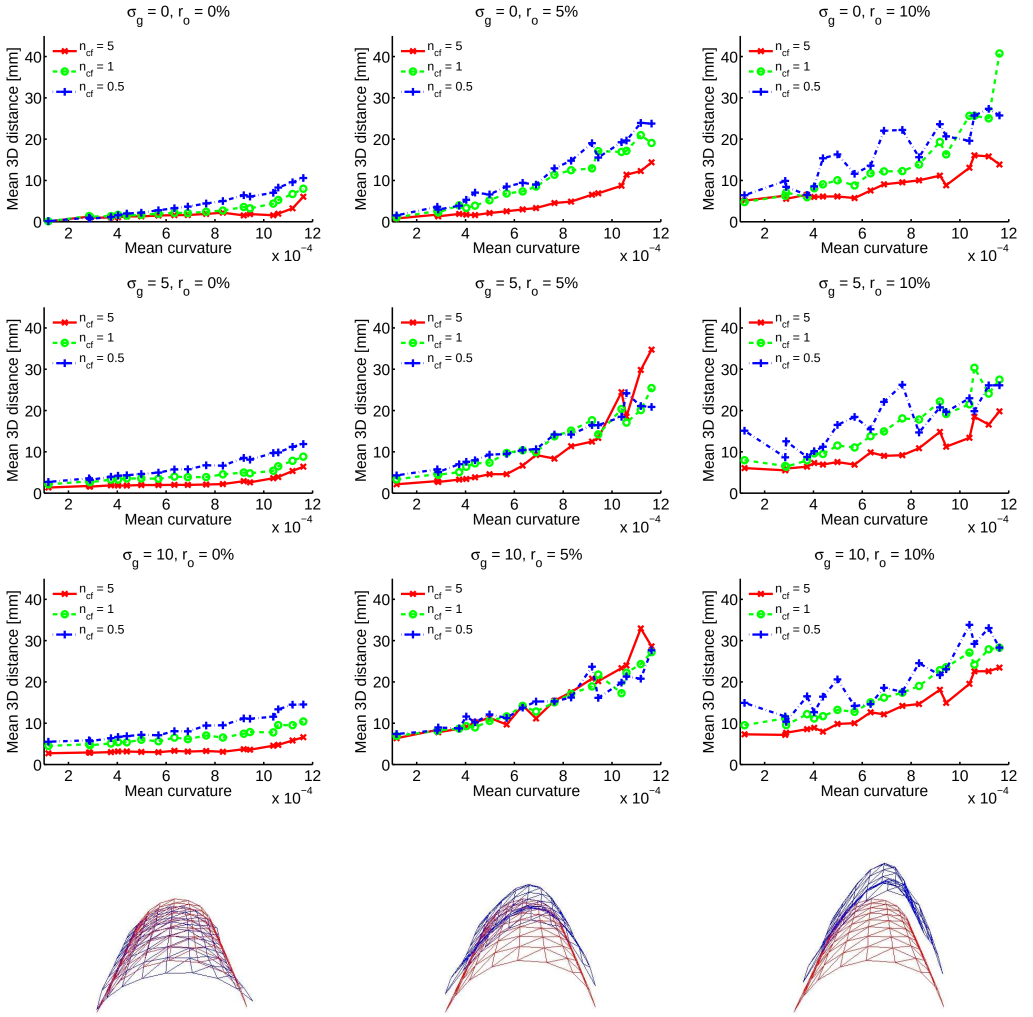
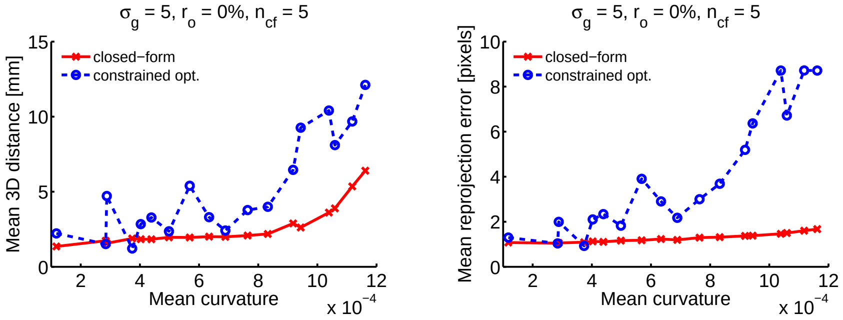


Figure 5.22: Shape recovery of a 200×200mm synthetic mesh imaged by a virtual camera placed 20cm away from it. Each plot shows the mean vertex-to-vertex 3D distance between the recovered surface and the ground-truth as a function of its mean curvature. The three different curves in each graph correspond to a varying number of correspondences per facet. Left to right, the number of outliers grows. Top to bottom, the gaussian noise added to the correspondences increases. For each experiments, we plot the average over 40 trials. The last row shows in blue recovered shapes for the ground-truth surface of Fig. 5.20(a,b), shown in red. The corresponding mean vertex-to-vertex distances are 9mm, 19mm and 38mm. This highlights the fact that even for distances around 40mm, the recovered shape remains meaningful.



(a) (b)

Figure 5.23: Comparison of our closed-form results against the results of constrained optimization. Optimization was performed on the vertex coordinates using Matlab's `fmincon` function, and starting from the flat position. (a) Mean vertex-to-vertex distance. (b) Reprojection error. Constrained optimization is both much slower and far less accurate than our approach.

5.5.1 Synthetic Data

We first applied our method to images such as those of Fig. 5.20(a), synthesized by projecting known deformed shapes using a virtual camera. The deformed shapes were obtained by recovering the 3D locations of reflective markers stuck on a 200×200mm piece of cardboard with an optical motion capture system. This allowed us to randomly create n_{cf} perfect correspondences per facet to which we added zero mean gaussian noise of variance σ_g. Finally, we simulated outliers by setting the image coordinates of r_o percents of the correspondences to uniformly and randomly distributed values.

In Fig. 5.22, we show results as a function of the surface's mean curvature, the maximum one being that of Fig. 5.20(a). Each plot includes three curves corresponding to $n_{cf} = \{5, 1, 1/2\}$, which depict the mean vertex-to-vertex 3D distance between the recovered mesh and ground-truth. The plots are ordered on a grid whose x-direction corresponds to $r_o = \{0\%, 5\%, 10\%\}$ and y-direction to $\sigma_g = \{0, 5, 10\}$. Each experiment was repeated 40 times, and we show the average results. Note that the error grows with the mean curvature of the shape, which is natural since the shape becomes more ambiguous when seen from the viewpoint shown in Fig. 5.20(a). In the last row, we display three shapes reconstructed from the image of Fig. 5.20(a) with their corresponding ground-truth. Note that even for average distances of 40mm between the true and recovered shape, the latter remains meaningful and could be used to initialize an iterative algorithm.

In Fig. 5.23, we compare our results against results obtained with Matlab's constrained optimization `fmincon` function. We used it to minimize the residual of the linear system of Eq. 5.11 with respect to the vertex coordinates, under the constraints that edge lengths must remain constant. We first tried to use the similar representation in terms of modes. However, since the constraints could never be truly satisfied, the algorithm would never converge towards an acceptable solution. This forced us to directly use the vertex coor-

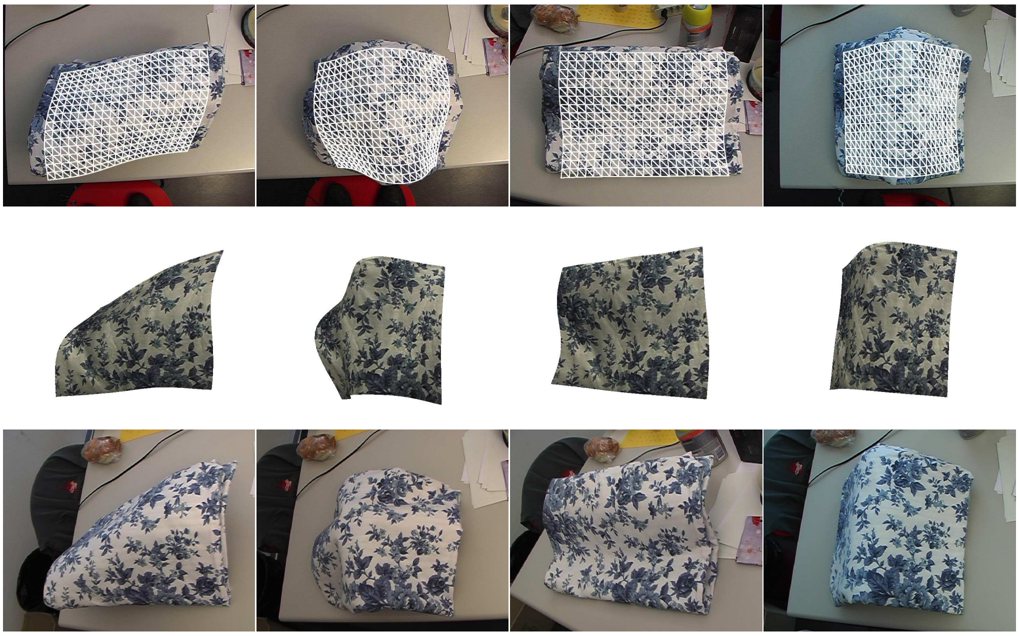


Figure 5.24: 3D registration of a folded bed-sheet to an individual image given a reference configuration. Top row: Recovered mesh overlaid on the original image. Middle row: Synthesized textured view using the recovered shape. Bottom row: Real side view of the sheet from similar viewpoints. Despite lighting changes, the synthetic images closely match the real ones.

dinates. To improve convergence and prevent the surface from crumpling, we added a smoothness term [127]. For all the frames, the initialization was set to the flat position. In Fig. 5.23(a), we show the mean 3D vertex-to-vertex distance for the case where $\sigma_g = 5$, $r_o = 0$, and $n_{cf} = 5$. The red curve corresponds to our closed-form solution and the blue one to constrained optimization. Note that our approach gives much better results. Furthermore, it is also much faster, requiring only 1.5 minutes per frame as opposed to 1.5 hours for constrained optimization. Fig. 5.23(b) shows the reprojection errors for the same cases.

5.5.2 Real Images

We tested our method on a folded bed-sheet, a piece of cloth and a t-shirt deforming in front of a 3-CCD DV-camera. In all these cases, we first established SIFT [103] correspondences between the reference image and the input one. We then applied the closed-form non-rigid 2D registration technique described in Section 3.3.1.3 to find the correct 2D projection of the mesh on the image. This let us establish dense 3D-to-2D matches. In the following results, we never explicily show 2D registration results. This would be meaningless, since they would look almost identical to the reconstructed 3D meshes reprojected onto the images that we show.

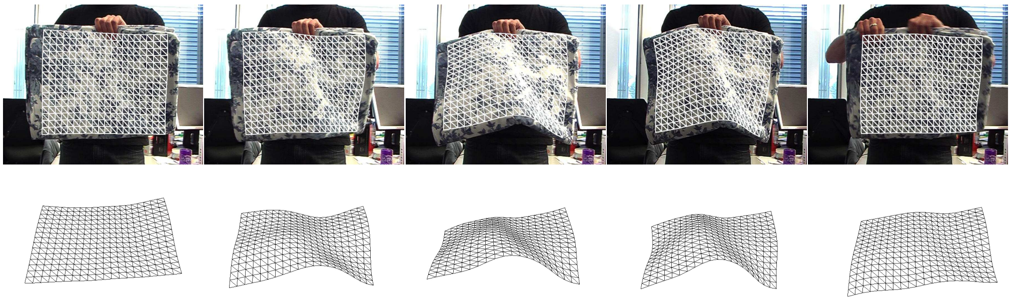


Figure 5.25: Shape recovery of a bed-sheet. Top row: Recovered mesh overlaid on the original image. Bottom row: Mesh seen from a different viewpoint.

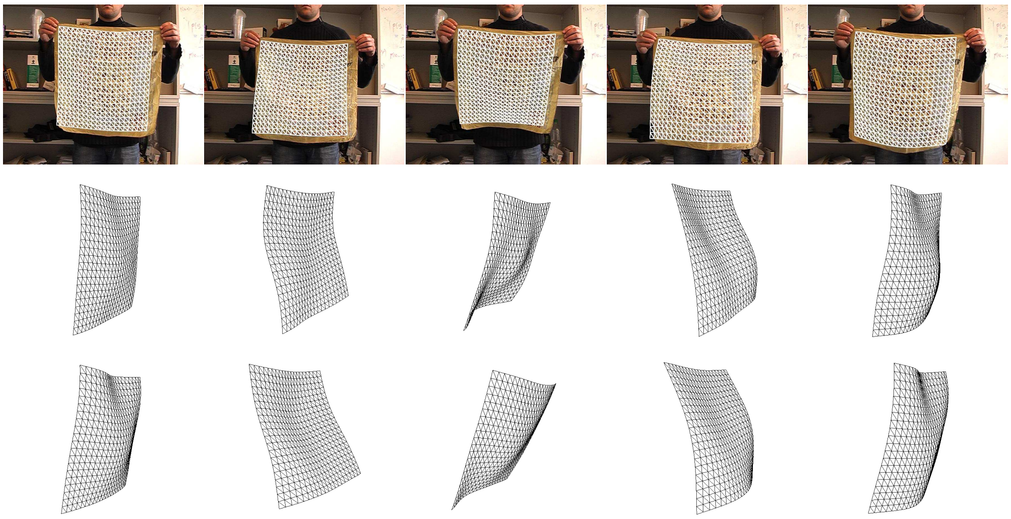


Figure 5.26: Shape recovery of a piece of cloth. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

In the case of the sheet, we deformed it into several unrelated shapes, took pictures from 2 different views for each deformation, and reconstructed the surface from a single image and a reference configuration. In Fig. 5.24, we show the results on four different cases. From our recovered shape, we generated synthetic textured images roughly corresponding to the viewpoint of the second image. As can be seen in the two bottom rows of Fig. 5.24, our synthetic images closely match the real side views. Additionally, we also reconstructed the same sheet from the images of a video sequence, and show the results in Fig. 5.25. Note that no initialization was required, and that nothing links one frame to the next.

In Figs. 5.26 and 5.27, we show results for images of a piece of cloth and of a t-shirt waved in front of the camera. Note that in both cases, the closed-form solution closely follows what we observe in the videos. To further refine it, we implemented a simple

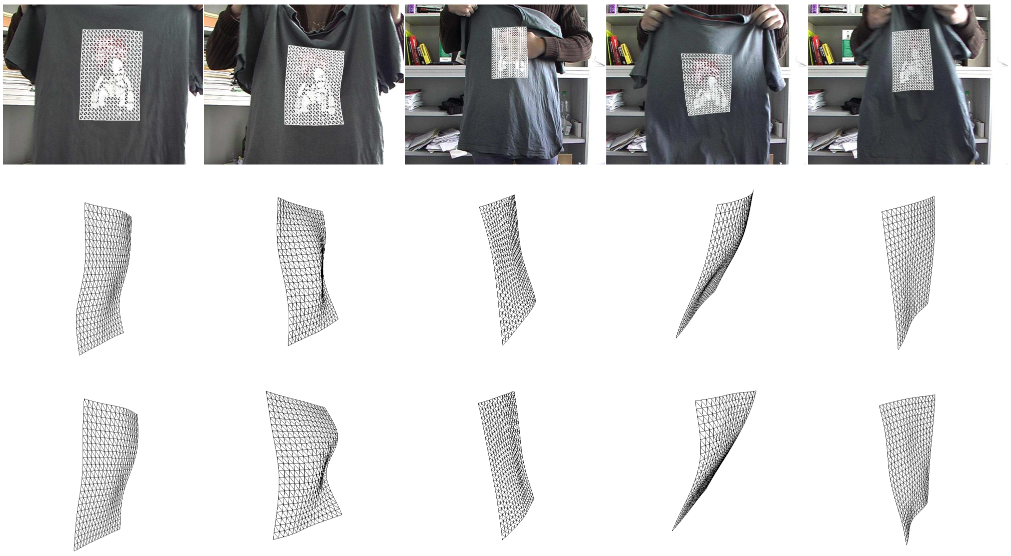


Figure 5.27: Shape recovery of the central part of a t-shirt. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

Gauss-Newton optimization technique, and minimize the residual $\|\tilde{\mathbf{D}}\tilde{\mathbf{b}} - \tilde{\mathbf{d}}\|$ corresponding to Eq. 5.22 with respect to the $\tilde{\beta}_i$. In the third row of the figures, we show the refined mesh after 5 iterations of this scheme. This proved sufficient to recover finer details at a negligible increase in overall computation time.

In some images of the t-shirt, such as those of Fig. 5.28, the motion blur was too large, or the drawing was too hidden to allow us to detect enough SIFT features. In such cases where we obtained less than 30 correspondences, we simply could not register the 2D mesh with the image, and therefore were unable to recover the shape of the surface. This, nonetheless, has no influence on the rest of the sequence, since we do not track, but detect the surface.

5.6 Conclusion

In this chapter, we have studied the use of global deformation models to recover the 3D shape of a non-rigid surface from images. We have first presented a way of automatically generating deformed versions of an inextensible mesh by varying randomly a determining subset of the angles between its facets. This let us create registered training examples from which we could build a linear deformation model using Principal Component Analysis. We then applied this low-dimensional model to reconstruct 3D surfaces from video using feature points and silhouettes. Finally, we showed that reconstruction could also be formulated as the closed-form solution to a set of quadratic equations. This alleviated the need of having a good initialization to the shape recovery process, and let us reconstruct non-rigid

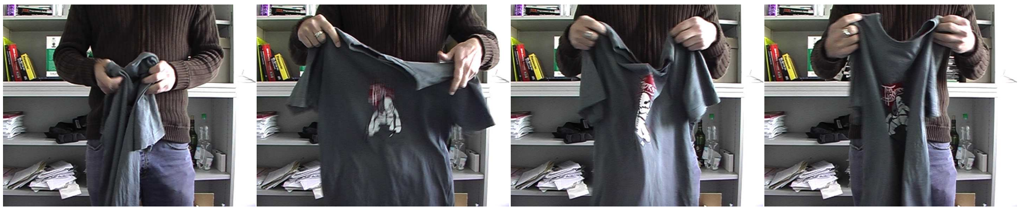


Figure 5.28: In such cases where a large part of the drawing on the t-shirt is hidden, or where the image becomes too blurry, not enough feature points could be found. We therefore fixed a threshold, and only recovered the shape in images where we found a least 30 SIFT correspondences. Note that, since we are not tracking the surface, this does not prevent us from correctly recover the shape in the other frames.

surfaces from individual images.

Even though the approach proposed in this chapter proved successful in many cases, it still suffers from several weaknesses. First, the manifold of the possible deformations of a highly flexible material is far from being linear. Therefore, PCA might not be the best choice to model it, as has already been noticed when PCA proved unable to respect inextensibility constraints. Second, global models may not always have enough degrees of freedom to model the complex deformations of a highly flexible surface. Finally, and most importantly, global deformation models are only valid for a specific surface. Therefore, a new model must be built for every new object's shape, even when it is made of a material seen before. In the next chapter, we will overcome these issues by introducing local models that represent the deformations of surface patches and can be combined to form arbitrary global shapes.

6 Local Deformation Models

In the previous chapter, we showed that global deformation models can effectively disambiguate the reconstruction problem. However, they do not always have enough degrees of freedom to handle complex deformations, such as those shown in Fig. 6.1(a). More importantly, a global model is only valid for a specific shape, and cannot be re-used for a different one, even if it is made of the same material, as illustrated in Fig. 6.1(b).

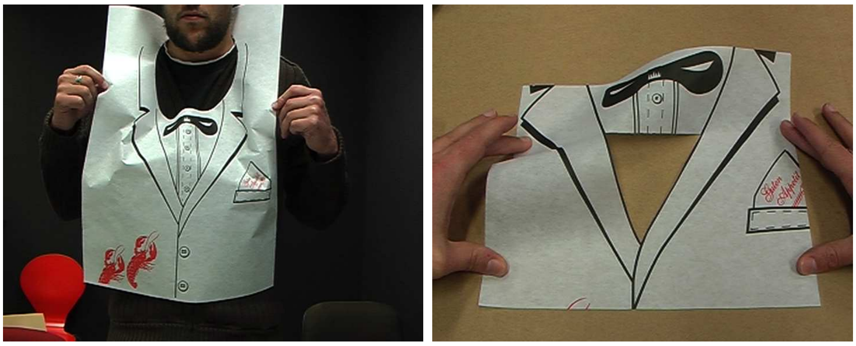
In this chapter, we show that we can replace the global deformation models by more flexible local ones. Given a surface made of a homogeneous material, we learn a model that approximates the behavior of a patch of such material, and replicate it to cover the whole object of interest. This lets us use the same local models to reconstruct the deformations of surfaces of arbitrary shapes, as long as they are made of the same material. Furthermore, since a small patch can only undergo much simpler deformations than a large object, much fewer examples are required to train the model.

A direct extension of the approach presented in the previous chapter is to model the deformations of a surface patch as linear combinations of modes. While effective for shape recovery, as in the global case, the linear local models do not account properly for properties such as surface inextensibility. One way to overcome this difficulty is to explicitly introduce inextensibility constraints into our algorithms. Another is to replace the linear models with non-linear ones. In this chapter, we explore both approaches.

More specifically, we learn local representations as Gaussian Process Latent Variable Models (GPLVM) [94]. This lets us build both linear and non-linear local models depending on the kernel we use. We then show that going from local to global can be done following a Product of Experts (PoE) paradigm [64]. This yields models that not only disambiguate the reconstruction and allow for closed-form solutions, as the global ones presented in the previous chapter, but also generalize to surfaces of arbitrary shape.

6.1 Learning Local Models

We now show how to learn our local deformation models. As in the global case, this requires training examples that represent the possible deformations of the surface of interest. However, rather than a global surface, these training examples now are much smaller surface patches. In theory, any technique that provides a density function over the high-dimensional space of the training examples would yield a suitable model. However, since the vertices of the mesh representation of surface patches cannot move independentely, the degrees of freedom of the training examples are coupled, and thus they lie on a low-dimensional manifold. We therefore learn models whose probability density functions are



(a) (b)

Figure 6.1: Advantages of the local models over global ones. (a) Highly flexible surfaces may undergo too complex deformations for the global models, whereas, locally, these deformations remain relatively simple. (b) A new global model must be learned for this surface even though it is made of the same material as the surface in (a). Local models can be combined into surfaces of arbitrary shapes.

conditioned on a space of reduced dimensionality. This alleviates the need of a number of training examples that grows exponentially with the dimensionality.

In the remainder of this section, we first propose a linear formulation of the local models, where the deformation of a patch is represented as a linear combination of modes. Since this formulation does not perfectly model the manifold of patch deformations, we replace it with one that more accurately accounts for the non-linearities of such a space. We then show that the linear models can be seen as a special case of the non-linear ones.

6.1.1 Linear Local Models

In the previous chapter, we have introduced global models where the shape of a surface was computed as a linear combination of modes. Such modes were obtained by applying PCA on a collection of synthetic training examples generated by randomly sampling a set of determining angles between the facets of a mesh.

Since these models proved effective at disambiguating 3D reconstruction, it seems natural to follow a similar idea for our local models. Given a set of N deformed surface patches $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T$ obtained, for example, in the same manner as for the global case, deformation modes can be taken as the eigenvectors $\mathbf{s}_i$ of the data covariance matrix $\mathbf{C} = \mathbf{Y}^T\mathbf{Y}$.

Under this formalism, the shape of a new patch $\mathbf{y}'$ can be expressed as

$$\mathbf{y}' = \mathbf{y}_0 + \mathbf{S}\mathbf{x}' \,, \tag{6.1}$$

where $\mathbf{y}_0$ is the mean shape, $\mathbf{S}$ is the matrix whose columns are the $\mathbf{s}_i$'s, and $\mathbf{x}'$ contains the weights of the different modes. The number of modes used in this representation is chosen so as to obtain the desired reconstruction accuracy.

As was already observed with the global models, and will be shown in the experimental results of this chapter, such a linear formulation does not yield an accurate representation of the surface deformations manifold. Indeed, such a space is non-linear and a linear technique only retrieves its englobing ellipsoid. Thus, for example, inextensibility constraints are not enforced by this linear formulation. To overcome this weakness, we therefore study non-linear local models. In theory, this could also have been done for global models. However, in practice, as will be shown in Chapter 7, learning a non-linear global model is intractable due to the complexity of the possible deformations and the number of required training examples.

6.1.2 The Gaussian Process Latent Variable Model

We learn our non-linear local deformation models as Gaussian Process Latent Variable Models (GPLVM) [94], which have been shown to be effective at recovering the underlying low-dimensional structure of high-dimensional data [161, 117]. The GPLVM relates a high-dimensional data set, $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T$, where $\mathbf{y}_i \in \Re^D$, and a low dimensional latent space, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T$, where $\mathbf{x}_i \in \Re^d$, using a Gaussian process mapping from $\mathbf{X}$ to $\mathbf{Y}$. The likelihood of the data given the latent positions can be expressed as

$$p(\mathbf{Y} \,|\, \mathbf{X}, \Theta) = \frac{1}{\sqrt{(2\pi)^{ND}|\mathbf{K}|^D}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right), \tag{6.2}$$

where the elements of the kernel matrix $\mathbf{K}$ are defined by a covariance function, k, such that $(\mathbf{K})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, which is entirely determined by the kernel hyper-parameters Θ.

The kernel matrix must be positive definite, but can be either linear or non-linear. Whereas there exists a single formulation of the linear kernel, as detailed in the following section, different non-linear covariance matrices can be chosen. In this work, we use a non-linear kernel that is the sum of a Radial Basis Function, a bias or constant term, and a noise term, and can be written as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Theta_1 \exp\left(-\Theta_2\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) + \Theta_3 + \Theta_4\delta_{i,j} , \tag{6.3}$$

where $\delta_{i,j}$ is the Kronecker delta function, and $\Theta_{l\ ,\ 1\leq l\leq 4}$ are the kernel hyperparameters.

Learning a GPLVM [94] involves maximizing the posterior

$$p(\mathbf{X}, \Theta \,|\, \mathbf{Y}) \ \propto\ p(\mathbf{Y} \,|\, \mathbf{X}, \Theta)\, p(\mathbf{X})\, p(\Theta) \tag{6.4}$$

with respect to $\mathbf{X}$ and Θ, where $p(\Theta)$ is a simple prior over the hyper-parameters of the covariance function, and $p(\mathbf{X})$ encourages the latent positions to be close to the origin.

Given a new test point $\mathbf{y}'$, inference in the GPLVM is done by maximizing $p(\mathbf{y}', \mathbf{x}'|\mathbf{X}, \mathbf{Y}, \Theta)$ with respect to the latent coordinates $\mathbf{x}'$ of the test point, or equivalently by minimizing its negative log likelihood given, up to an additive constant, as

$$\mathcal{L}_{local}(\mathbf{x}', \mathbf{y}') = \frac{\|\mathbf{y}' - \mu(\mathbf{x}')\|^2}{2\sigma^2(\mathbf{x}')} + \frac{D}{2}\ln\sigma^2(\mathbf{x}') + \frac{1}{2}\|\mathbf{x}'\|^2 , \tag{6.5}$$

with mean and variance given by

$$\mu(\mathbf{x}') = \mathbf{y}_0 + \mathbf{Y}^T\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}') \,, \tag{6.6}$$

$$\sigma^2(\mathbf{x}') = k(\mathbf{x}',\mathbf{x}') - \mathbf{k}(\mathbf{x}')^T\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}') \,, \tag{6.7}$$

where $\mathbf{y}_0$ is the original mean value of the data, and $\mathbf{k}(\mathbf{x}')$ is the vector with elements $k(\mathbf{x}', \mathbf{x}_j)$ for latent positions $\mathbf{x}_j \in \mathbf{X}$.

While effective at learning complex manifolds, the GPLVM suffers from the fact that its computional cost grows as $\mathcal{O}(N^3)$, due to the inversion of the $N \times N$ kernel matrix. Sophisticated sparsification techniques have recently been proposed [96] to overcome this issue. Such techniques introduce a set of m inducing variables $\mathbf{X_u}$, which allows to split the covariance matrix into two matrices $\mathbf{K_{u,u}}$ and $\mathbf{K_{f,u}}$. The former is the kernel matrix for the elements of $\mathbf{X_u}$, and the latter denotes the covariance between $\mathbf{X}$ and $\mathbf{X_u}$. This reduces the computational complexity of learning to $\mathcal{O}(Nm^2)$, and has proved more accurate than simply using a subset of the data. In our particular case, we use the Fully Independent Training Conditional (FITC) approximation [143], which involves an independence assumption when estimating the probability of the examples given the inducing variables.

In this sparse formulation, learning is done by maximizing the posterior

$$p(\mathbf{Y} \mid \mathbf{X}, \mathbf{X_u}, \Theta) = \mathcal{N}\left(\mathbf{K_{f,u}}\mathbf{K_{u,u}^{-1}}\mathbf{X_u}, diag[\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}] + \beta^{-1}\mathbf{I}\right) \tag{6.8}$$

with respect to $\mathbf{X}$, $\mathbf{X_u}$ and Θ, where $diag[\mathbf{B}]$ is a diagonal matrix whose elements match the diagonal of $\mathbf{B}$, and $\mathbf{Q_{f,f}} = \mathbf{K_{f,u}}\mathbf{K_{u,u}^{-1}}\mathbf{K_{u,f}}$.

Finally, for inference, we can re-write the mean and variance of the sparse GPLVM likelihood $p(\mathbf{y}', \mathbf{x}' | \mathbf{X_u}, \mathbf{Y}, \Theta)$ as

$$\mu_s(\mathbf{x}') = \mathbf{y}_0 + \mathbf{Y}^T\mathbf{K_{u,f}}\mathbf{A}^{-1}\mathbf{k_u} \,, \tag{6.9}$$

$$\sigma_s^2(\mathbf{x}') = k(\mathbf{x}',\mathbf{x}') - \mathbf{k_u}^T(\mathbf{K_{u,u}^{-1}} - \beta^{-1}\mathbf{A}^{-1})\mathbf{k_u} \,, \tag{6.10}$$

where $\mathbf{A} = \beta^{-1}\mathbf{K_{u,u}} + \mathbf{K_{u,f}}\mathbf{K_{f,u}}$, and $\mathbf{k_u}$ is the vector with elements $k(\mathbf{x}', \mathbf{x}_j)$ for latent positions $\mathbf{x}_j \in \mathbf{X_u}$.

When used for tracking purposes [161], the likelihood of the GPLVM, sparse or not, is typically optimized with respect to $\mathbf{y}'$ and $\mathbf{x}'$. This is in contrast with the linear approach proposed in Section 6.1.1, where $\mathbf{y}'$ is directly taken as the mean prediction of the model. The reason for optimizing both variables is that it allows the model to better generalize and avoid remaining too close from the training examples. Furthermore, this will prove crucial to combine local models into a global one.

6.1.3 Probabilistic PCA as a GPLVM

We now show that the linear model introduced in Section 6.1.1 is a particular case of the GPLVM, and thus that the framework introduced in the previous section is valid for both linear and non-linear representations. Given a linear kernel, the GPLVM has been shown to be equivalent to probabilistic principal component analysis [95]. The covariance matrix then becomes

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I} \, , \tag{6.11}$$

where β is the inverse noise variance of the model.

When learning a GPLVM under this assumption, the maximum of the likelihood of Eq. 6.2 can be computed analytically, and is reached for

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T \, , \tag{6.12}$$

where $\mathbf{U}$ is the $N \times d$ matrix containing the first d eigenvectors of $\mathbf{C}' = \mathbf{Y}\mathbf{Y}^T$. $\mathbf{L}$ is a similar matrix as in Eq. 5.21 whose diagonal elements are $(\lambda_i - \beta^{-1})^{-1/2}$ where λ_i's are the eigenvalues of $\mathbf{C}'$. Finally, $\mathbf{V}$ is an arbitrary rotation matrix.

This can be thought of as a dual formulation of probabilistic PCA, since, in its standard form, the high-dimensional data is related to its low-dimensional representation through the eigenvectors $\mathbf{s}_i$ of the data covariance matrix $\mathbf{C} = \mathbf{Y}^T\mathbf{Y}$. The eigenvectors of both formulations are linked through the relation

$$\mathbf{S} = \mathbf{Y}^T\mathbf{U}\Lambda^{-\frac{1}{2}} \, , \tag{6.13}$$

where $\mathbf{S}$ is the matrix whose columns are the $\mathbf{s}_i$'s, and their eigenvalues, contained in the diagonal matrix Λ, are the same.

For inference, the negative log likelihood of a new test point $\mathbf{y}'$ and its latent representation $\mathbf{x}'$ is still given by Eq. 6.5. Furthermore, by setting $\mathbf{V}$ to the identity matrix, we can use Eqs. 6.12 and 6.13 to rewrite the linear kernel of Eq. 6.11 as a function of $\mathbf{S}$, $\mathbf{Y}$, and Λ. Substituting this in the mean prediction of Eq. 6.6 lets us obtain, up to a scaling by Λ, the usual probabilistic PCA mean prediction

$$\mu_l(\mathbf{x}') = \mathbf{y}_0 + \mathbf{S}\mathbf{x}' \, . \tag{6.14}$$

Note that this corresponds to the linear formulation introduced in Section 6.1.1.

As mentioned in the previous section, for tracking purposes, the negative log likelihood of Eq. 6.5 is typically optimized with respect to $\mathbf{y}'$ and $\mathbf{x}'$ to generalize over the training examples. In the linear case, this can even be achieved by optimizing with respect to $\mathbf{y}'$ only, while still following the model. Since the $\mathbf{s}_i$'s are orthonormal, the latent variable $\mathbf{x}'$ corresponding to the test point can directly be obtained by projecting $\mathbf{y}'$ into the low-dimensional manifold, which we can write as

$$\mathbf{x}' = \mathbf{S}^T(\mathbf{y}' - \mathbf{y}_0) \, . \tag{6.15}$$

Eq. 6.14 then becomes

$$\mu_l(\mathbf{y}') = \mathbf{y}_0 + \mathbf{S}\mathbf{S}^T(\mathbf{y}' - \mathbf{y}_0) \, . \tag{6.16}$$

In this formulation, the latent representation $\mathbf{x}'$ does not appear anymore, since it can directly be computed from $\mathbf{y}'$. By contrast with the non-linear formulation, this therefore yields a model whose number of degrees of freedom only depends on the number of vertices of the mesh.

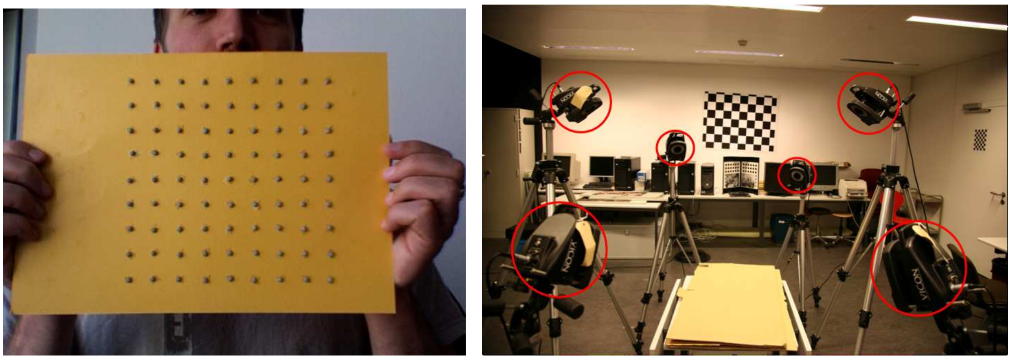


Figure 6.2: We used an optical motion capture system to acquire real training data. Left: We stuck reflective markers as a rectangular grid on the surface of interest. Right: We deformed the object in front of six infrared cameras.

6.1.4 Acquiring the Training Data

As we have seen in Sections 6.1.1 and 6.1.2, learning a local model requires training data. When dealing with large surfaces, the amount of necessary data to cover the space of possible deformations can be very large. However, since local patches have fewer degrees of freedom and can only undergo relatively small deformations, learning local deformation models becomes easier.

One way to create training examples would be to apply the technique of Section 5.1.2 to obtain deformed versions of the patches. This is perfectly suited for the linear formulation of Section 6.1.1, where the generated modes typically represent deformations sorted from low spatial frequencies to higher ones. Because the higher frequency modes can be dropped, the training data does not need to be particularly accurate. By contrast, a non-linear mapping interpolates more accurately between the training examples than the linear one, which makes synthetic data inappropriate, as shown in Chapter 7. In this case, it is important that the training examples be representative of the real behavior of the material of interest, and thus, real data is required.

6.1.4.1 Real Data Acquisition

To collect training examples from real surfaces, we used a Vicon$^{\mathrm{TM}}$ optical motion capture system. As depicted in Fig. 6.2, we stuck 3mm wide hemispherical reflective markers as a rectangular grid on a surface and deformed it arbitrarily in front of six infrared Vicon$^{\mathrm{TM}}$ cameras that reconstruct the 3D positions of individual markers. Since the markers were positioned to form a $P \times Q$ grid, let $\tilde{\mathbf{y}} = [x_1, y_1, z_1, ..., x_{P\times Q}, y_{P\times Q}, z_{P\times Q}]^T$ be the vector of their concatenated coordinates acquired at a specific time. Our goal being to learn a local model, as opposed to a global one, we decomposed $\tilde{\mathbf{y}}$ into overlapping $p \times q$ rectangular patches centered on individual grid vertices, as shown in Fig. 6.3.

We collected these patches from individual frames in several motion sequences, sub-

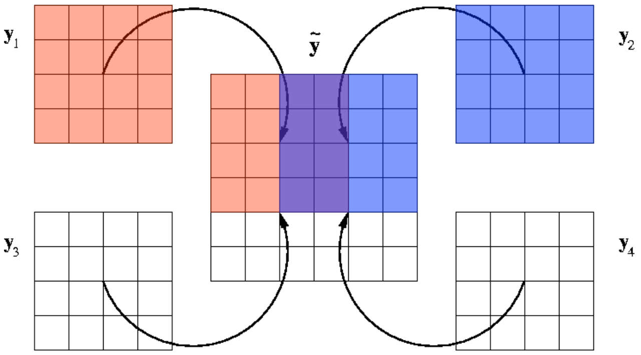


Figure 6.3: Decomposing the surface into patches. In this case, the global surface $\tilde{\mathbf{y}}$ is composed of four overlapping patches $\mathbf{y}_{1,..,4}$.

tracted their mean, and symmetrized them with respect to their x-, y- and z-axes to obtain additional examples. This resulted in a large set of N_p $p \times q$ patches $\mathbf{y}_{i\ ,\ i=1,..,N_p}$. Since the sequences were acquired at 30 Hz, they comprised many similar deformations that do not bring new information. We therefore retained only a subset $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]^T$ of $N < N_p$ patches that were different enough from each other based on a vertex-to-vertex distance criterion.

In particular, we used this technique for two different materials: Relatively rigid cardboard and a more flexible tissue paper. For the cardboard, we placed the reflective markers on a 9×9 grid, and for the napkin, in a 9×7 one. This difference in resolution was only introduced to facilitate the motion capture and has no bearing on the rest of the approach. In both cases, the markers were placed 2cm apart in both directions. Out of 10 motion sequences for each material, we set one aside for validation purposes and used the other 9 for learning. In each frame, we selected five 5×5 patches for the cardboard and six for the napkin, and pruned the resulting set such that the minimum distance between corresponding vertices in separate patches was greater than 0.7cm for the cardboard and 1cm for the napkin. This produced 2032 patches for the cardboard and 2881 for the napkin. The larger number of the latter reflects the greater flexibility of the tissue paper.

6.2 Global Models as Mixtures of Local Ones

Our ultimate goal is to recover the shape of a surface from images. We therefore need a global model of the surface of interest. Rather than learning a global representation for a particular surface, which would imply doing so for every new one, we make use of the local models introduced in Section 6.1 and combine them using a Product of Experts (PoE) [64] paradigm. The choice of PoE is a natural one, since they were explicitly designed to model high-dimensional data subject to low-dimensional constraints, as is the case for a large surface made of small patches. We represent a global surface as a triangulated mesh with N_v vertices $\mathbf{v}_i = [x_i, y_i, z_i]^T$, $1 \leq i \leq N_v$ connected by N_e edges. We define $\tilde{\mathbf{y}}' =$

$[\mathbf{v}_1^T, \cdots, \mathbf{v}_{N_v}^T]^T$ as the vector of 3D coordinates obtained by concatenating the vertices $\mathbf{v}_i$.

6.2.1 PoE for Deformable Surfaces

As mentioned above, PoE [64] are good at representing high-dimensional data subject to low-dimensional constraints by combining probabilistic models. Each constraint is treated by an individual expert, which gives a high probability to the examples that satisfy it. The probability of examples statisfying some constraints but violating others will naturally be zeroed out by the experts associated with the violated constraints.

In the general case, training a PoE is difficult because one has to identify the experts that simultaneously maximize the probabilities of the training examples and assign low probabilities to unobserved regions of the data space. However, in the case of homogeneous surfaces, this task is greatly simplified; We do not need to identify the different experts since all local patches obey the same deformation rules, i.e. all experts are the same. As a consequence, one can simply train a single local deformation model corresponding to one expert and, for inference, replicate it to cover the entire surface as shown in Fig. 6.3. This simply assumes that maximizing the likelihood of a global shape is achieved through maximizing the likelihoods of all the patches. Note that the choice of the patch size influences both the local and global representations. Smaller sizes result in local models that are more constrained since less deformations are possible, but impose a higher number of experts to cover the global surface. Furthermore, we use overlapping patches to enforce smooth transitions between neighboring experts. However this does *not* impose global surface smoothness, since the local models may allow for sharp folds.

More formally, let $\mathbf{y}'_i$, $1 \leq i \leq S$ be the vectors of the 3D coordinates of the S overlapping patches associated with the experts, where $\mathbf{y}'_i \subset \tilde{\mathbf{y}}'$. Note that, because patches overlap, the same vertex coordinates of $\tilde{\mathbf{y}}'$ appear in several $\mathbf{y}'_i$'s. Furthermore, note also that using a single global mesh $\tilde{\mathbf{y}}'$ prevents conflicts arising from two experts predicting different shapes for the same mesh protion. Let $\mathbf{x}' = \left[\mathbf{x}'^T_1, ..., \mathbf{x}'^T_S\right]^T$ be the low-dimensional latent coordinates of each of the S experts. Under our formalism, the conditional probability of the global surface can be expressed as

$$p(\tilde{\mathbf{y}}'|\mathbf{x}', \mathcal{M}) = \frac{\prod_i p_i(\mathbf{y}'_i|\mathbf{x}'_i, \mathcal{M})}{\int \prod_i p_i(\mathbf{y}'_i|\mathbf{x}'_i, \mathcal{M}) d\tilde{\mathbf{y}}'} , \tag{6.17}$$

where $\mathcal{M}$ is a local model described in Section 6.1.

Since the denominator of Eq. 6.17 is constant, we can define a prior over the deformation of the whole surface according to all the experts whose negative log is

$$\mathcal{L}_{poe}(\mathbf{x}', \tilde{\mathbf{y}}') = \sum_{i=1}^{S} \mathcal{L}_{local}(\mathbf{x}'_i, \mathbf{y}'_i) , \tag{6.18}$$

where $\mathcal{L}_{local}$ is the local negative log likelihood defined in Eq. 6.5.

6.2.2 Surface Boundary Effects

As can be observed in Fig. 6.3, vertices in the center of the surface appear in several patches, whereas those in the corners belong to a single one. As a consequence, the latter have less influence on the global negative log likelihood given by Eq. 6.18. To prevent them from moving freely, we re-weight the vertices such that their influence is inversely proportional to the number of patches they belong to. This is done by replacing in Eq. 6.5 the distance between the patch vertices $\mathbf{y}'_i$ and their mean prediction $\mu(\mathbf{x}'_i)$, by the term

$$\Delta(\mathbf{x}'_i, \mathbf{y}'_i) = \sum_{j=1}^{p \times q} \frac{1}{V(i,j)} (\mathbf{W}\mathbf{y}'_{i,j} - \mu^j(\mathbf{x}'_i))^2 \ , \tag{6.19}$$

where $\mathbf{y}'_{i,j}$ is the j^{th} vertex of patch i and $\mu^j(\mathbf{x}'_i)$ its corresponding mean prediction. $V(i,j)$ is the number of patches for a vertex, which depends on the index in the global representation of the j^{th} vertex of patch i. Furthermore, we also introduced a 3×3 diagonal matrix $\mathbf{W}$ in Eq. 6.20 that defines the global scales along the x-, y- and z-axes, and accounts for the difference in scale between the training and testing surfaces. In practice, we allow for at most 10% scaling. The negative log likelihood of the global surface can then be written

$$\mathcal{L}_{global}(\mathbf{x}', \tilde{\mathbf{y}}') = \sum_{i=1}^{S} \left(\frac{\Delta(\mathbf{x}'_i, \mathbf{y}'_i)}{2\sigma^2(\mathbf{x}'_i)} + \frac{D}{2} \ln \sigma^2(\mathbf{x}'_i) + \frac{1}{2} \|\mathbf{x}'_i\|^2 \right) \ . \tag{6.20}$$

6.3 Monocular 3D Tracking

Here, we formulate our reconstruction algorithm as a frame-to-frame optimization problem. Even though tracking the surface deformations from one frame to the next increases the risk of drifting, it might be the only practical solution in some cases, for example when the surface is not sufficiently textured to rely on feature points. In such cases, template matching is more adapted, but yields an objective function with local minima that requires a good initialization. Similarly, the negative log likelihood of the non-linear local model is a highly non-convex function and needs a good starting point to converge to a correct solution, which makes tracking more suitable.

At each time t, we seek to recover a state vector ϕ_t that determines the shape of the surface. With the non-linear local models, the state is defined as $\phi_t = \left[\tilde{\mathbf{y}}_t'^T, \mathbf{x}_t'^T\right]^T$, where $\tilde{\mathbf{y}}'_t$ is the vector of the 3D coordinates of the global surface, and $\mathbf{x}'_t = \left[\mathbf{x}_{1,t}'^T, ..., \mathbf{x}_{S,t}'^T\right]^T$ denotes the latent variables for the local models. With the linear models, $\phi_t = \tilde{\mathbf{y}}'_t$, because, as mentioned in Section 6.1.3, the latent representations can be obtained directly from $\tilde{\mathbf{y}}'_t$. Note that both formulations guarantee surface continuity, since the patches share a common vector of vertex coordinates. This would not have been the case if we had optimized with respect to the latent variables only, since several experts can predict different locations for the same vertex. Given an image $\mathbf{I}_t$ and a local deformation model $\mathcal{M}$, we look for the MAP estimate ϕ_t, and therefore approximate the posterior

$$p(\phi_t|\mathbf{I}_t, \mathcal{M}) \propto p(\mathbf{I}_t|\phi_t)p(\phi_t|\mathcal{M}) \ , \tag{6.21}$$

where $p(\mathbf{I}_t|\phi_t)$ is the image likelihood, and the negative log of $p(\phi_t|\mathcal{M})$ is given by Eq. 6.20.

6.3.1 Image Likelihood

To estimate the image likelihood, we rely on texture and edge information. The latter constrains the boundary vertices which are not as well-constrained by texture as the interior ones. We assume that both sources of information are independent given the state, which writes

$$p(\mathbf{I}_t|\phi_t) = p(\mathbf{T}_t|\phi_t)p(\mathbf{E}_t|\phi_t) \ . \tag{6.22}$$

To take advantage of the whole texture, we use template matching. The negative log likelihood of such an observation is given by

$$-\ln\ p(\mathbf{T}_t|\phi_t) = \frac{1}{\sigma_T^2} E_{TM}(\tilde{\mathbf{y}}') \ , \tag{6.23}$$

where E_{TM} was defined in Eq. 3.11, and σ_T is a constant set to the variance of the expected texture error.

To constrain the boundary of the surface, we first project the border of the mesh into the image. We then sample points $\mathbf{e}_i$ on this projected boundary, and look in the direction of their normal for edge points $\mathbf{u}_{i,j}$ detected by Canny's algorithm. We allow for multiple hypotheses and retain all the matches within a distance r from the current reprojection. Starting from 8 pixels, this distance is iteratively divided by 2 after a fixed number of optimization steps, until it reaches 2 pixels. The negative log likelihood of the edge observations is then

$$-\ln\ p(\mathbf{E}_t|\phi_t) = \frac{1}{\sigma_E^2} \left(\frac{1}{r^2} \sum_{i=1}^{N_s} \sum_{j=1}^{N_h(i)} \|\mathbf{u}_{i,j} - \mathbf{e}_i(\phi_t)\|^2 \right) \ , \tag{6.24}$$

where N_s is the number of sampled boundary points, and $N_h(i)$ is the number of edge hypotheses for point i. As for texture, σ_E is a constant corresponding to the variance of the expected error. In practice, our method is relatively insensitive to the exact values of σ_T and σ_E.

6.3.2 Optimization

Reconstruction is performed by minimizing the negative log of the approximate posterior of Eq. 6.21, which we write, up to an additive constant, as

$$\mathcal{L}_{tot}(\phi_t) = \mathcal{L}_{global}(\phi_t) - \ln\ p(\mathbf{T}_t|\phi_t) - \ln\ p(\mathbf{E}_t|\phi_t) \ . \tag{6.25}$$

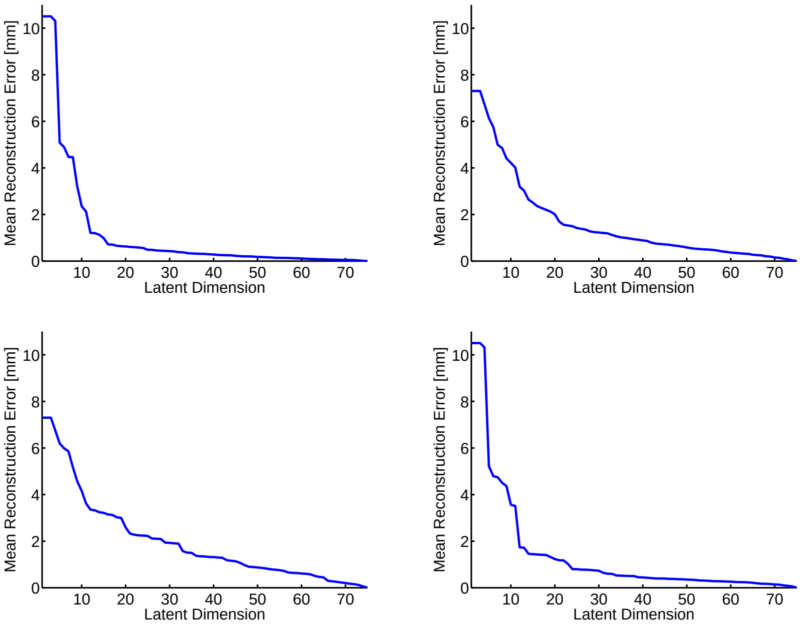


Figure 6.4: Validating the linear deformation models. We compute the mean of the average vertex-to-vertex distances between test data and the model predictions, and plot it versus the number of modes. Top: The error for the cardboard model (left) decreases faster than for the napkin one (right). This corresponds to our intuition that fewer dimensions are necessary to model the deformations of a more rigid material. Bottom: We tried to reconstruct the napkin data using the cardboard model (left), as well as the opposite (right). We can observe that, since the napkin is more flexible, its deformations are a superset of those of the cardboard, and thus can model it quite accurately. This is not the case when trying to reconstruct napkin data with the cardboard model.

In practice, we assume that the camera projection matrix is known and remains constant throughout the sequence. This entails no loss of generality since the vertices are free to move rigidly.

In the first frame of a sequence, we start from the reference shape and, in the non-linear case, initialize the latent positions of the local models such that their mean predictions best correspond to the different patches of the reference shape. This is done by optimizing the negative log likelihood of Eq. 6.5. Then, at every frame, we initialize the state with the MAP estimate of the previous time and optimize to get the new shape.

When considering large surfaces, the number of degrees of freedom of our optimization problem quickly becomes large, since it includes the 3D positions of the vertices. To improve convergence, we introduce a coarse-to-fine approach to optimization. In the first step we only consider every other row and every other line of the grid representing the local patches. Therefore, we end up with patches of 3×3 vertices separated by 4cm instead of 5×5 vertices separated by 2cm. While not changing the number of local models that we use, this drastically reduces the number of vertices to optimize. Furthermore, this only changes the resolution of the patches, but not their size. Therefore we can still use the same local deformation models to represent the shape of the patches. This proved most useful with the non-linear models, since the resulting objective function is much more complex than in the linear case.

As mentioned in Section 6.1.1, using a linear local model does not enforce the inextensibility of the mesh. This is still the case when using a prior over the latent variables to prevent them from taking overly large values, since extension typically appears in the first 10 modes. To overcome this issue and prevent the mesh from stretching, we introduce a prior over the global shape $\tilde{\mathbf{y}}'$ whose negative log is

$$-\ln\, p(\tilde{\mathbf{y}}') = \frac{1}{\sigma_D^2} \sum_{i=1}^{N_v} \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \left(\|\mathbf{v}_i - \mathbf{v}_j\| - L_{i,j}\right)^2 , \tag{6.26}$$

where $\mathcal{N}(\mathbf{v}_i)$ is the set of neighbors of vertex $\mathbf{v}_i$, and σ_D corresponds to the variance of the expected extension error and has the same value for every sequence. When using a linear model, this term is simply added to Eq. 6.25. This prior could also be used in the non-linear case, since nothing explicitly enforces these constraints. However, as shown in Section 6.4, even though it yields better accuracy on synthetic results, it did not prove necessary to reconstruct surfaces from real sequences.

6.4 Experimental Results

In this section, we first validate the local models we learned for cardboard and tissue paper, and then use both synthetic and real data to demonstrate that they sufficiently constrain the reconstruction to achieve accurate results, even when the lack of texture on the surfaces makes it difficult for texture-based approaches.

6.4.1 Local Models Validation

We used the technique of Section 6.1 to learn models for the two datasets discussed in Section 6.1.4 for increasing latent dimensions. We then picked the dimensionality that best fitted our validation set.

In the linear case, this is done by computing the mean prediction given by Eq. 6.16 and taking the error as the average vertex-to-vertex distance between this prediction and the true test shape. In the top row of Fig. 6.4, we plot the mean of these values for each of the dimensions, ranging from 1 to 75. Note that, in the case of cardboard, the curve quickly decreases, thus indicating that using a small number of modes is sufficient to accurately represent a large portion of the shape space, whereas for the paper napkin, more modes are required. This tallies with our intuition that the manifold of potential deformations of

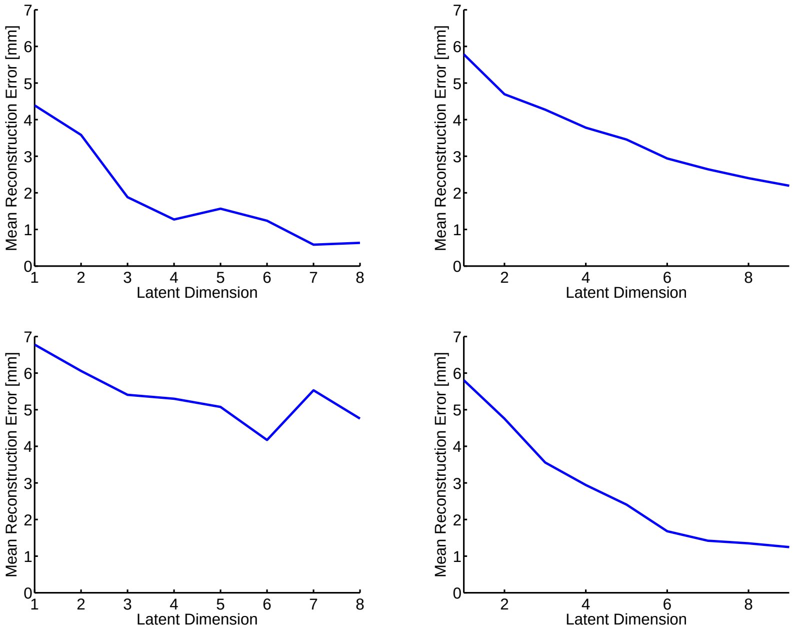


Figure 6.5: Validating the non-linear deformation models. The same reconstruction error as in the linear case is computed and plotted as a function of the latent dimension. Top: In the cardboard case (left), we chose latent dimension 4, since it corresponds to the point where the error stabilizes. In the case of the napkin (right), we chose to use dimension 7 because of convergence problems during training in dimensions 8 and 9. These would have required a larger number of inducing variables, which would have incerased the computational burden. Bottom: As in the linear case, we can observe that the cardboard models are unable to reconstruct napkin data, whereas the inverse is possible.

the napkin is larger than that of the cardboard. In the bottom row, we display, on the left, the same error when trying to reconstruct the napkin test data using the cardboard model, and, on the right, the opposite. As expected, the napkin model gives good reconstructions of cardboard data, since the deformations of cardboard are a subset of those of the paper napkin. The inverse is not true.

In the non-linear case, we picked latent dimensions ranging between 1 and 9. For each patch $\mathbf{y}'_i$ extracted from the validation sequence, we infered the corresponding latent variables $\mathbf{x}'_i$ by minimizing the negative log likelihood given in Eq. 6.5, and computed its mean prediction from Eq. 6.9. In Fig. 6.5, we plot the same curves as for the linear case. For the cardboard, the models were all trained using 100 inducing variables. We picked $d = 4$ since it corresponds to the point where the error stabilizes. For some larger values of d, we can even observe worse results, thus indicating that the model overfits the training data. For the napkin that has more samples and a greater variety of observed shapes, we had to use

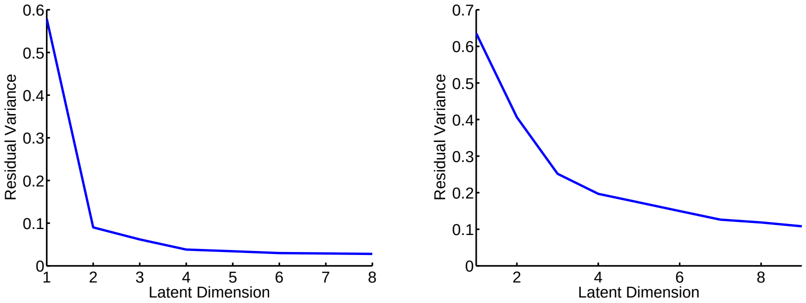


Figure 6.6: We used Isomap to compute the low-dimensional embeddings of our cardboard (left) and napkin (right) data for different latent dimensions. We plot the residual variances given by Isomap as a function of the dimension. This confirms our choice of dimension 4 for the cardboard and 7 for the napkin.

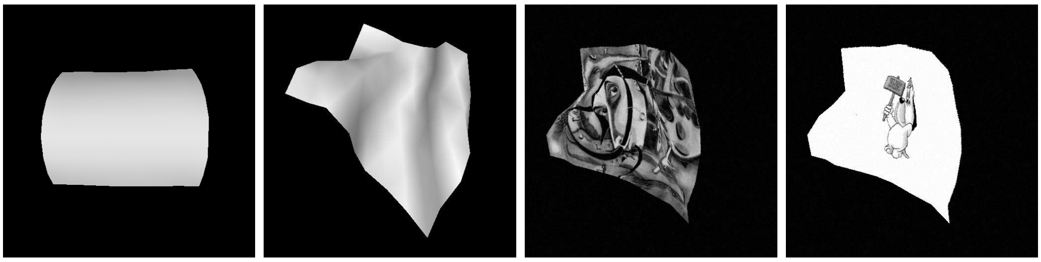

(a) (b) (c) (d)

Figure 6.7: Synthetic images generated from optical motion capture data (a) Shaded view of a cardboard surface (b) Similar shaded view for a paper napkin. (c,d) Images synthesized by texture-mapping using either a rich texture or a much more uniform one.

200 inducing variables to make the training process converge. In this case, the higher values of d yield slightly better results. However, training in dimensions 8 and 9 suffered from problems in convergence and would have required a larger number of inducing variables. Since this would imply a larger computational burden, in our experiments we used $d = 7$, which we will show to be sufficient for our purposes. In the bottom row of the figure, it can again be observed that the cardboard model is unable to correctly predict napkin shapes, whereas the inverse is possible.

Another way of finding the latent space dimension is to use Isomap [149] to unfold the high-dimensional shape space to a lower-dimensional manifold. This non-linear technique has proved efficient at finding the underlying dimension of a problem, but provides no inverse mapping or density over the shape space. It therefore cannot be used for inference, as required for tracking, but it allowed us to confirm our latent dimensions. As can be seen in Fig. 6.6, the residual variance given by Isomap also points to dimension 4 for the cardboard and 7 for the paper napkin.

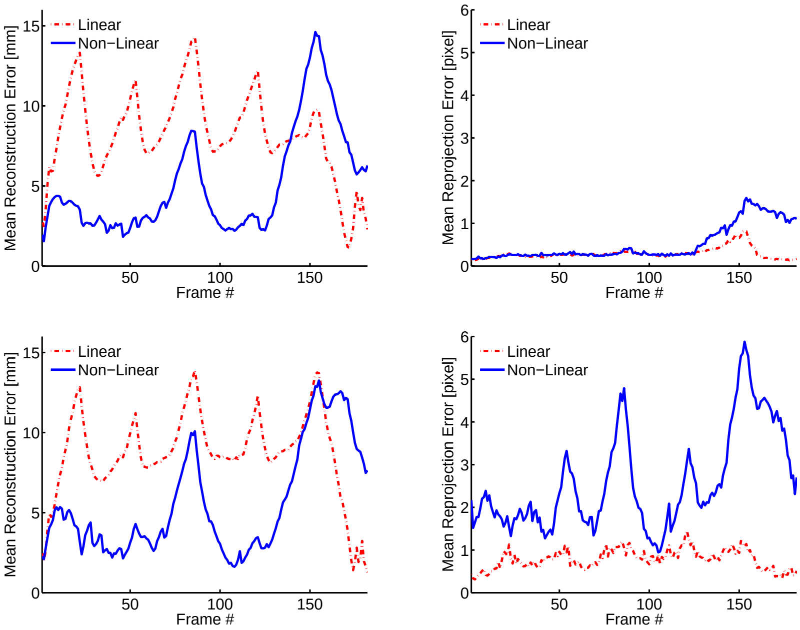


Figure 6.8: Comparison of the linear (dashed red) and non-linear (solid blue) models for cardboard using sequences of synthetic images. No penalization of extension was used to obtain these results. Top row: For each of the well-textured images, we plot, on the left, the mean 3D vertex-to-vertex distance, and, on the right, the mean reprojection error of randomly sampled surface points. Bottom row: Same plots for much less textured images. Note that the non-linear models yield a better 3D reconstruction than the linear ones. This is to be expected since they suffer less from the absence of stretching penalty.

6.4.2 Synthetic Data

We measured the accuracy of our method on synthetically generated images. Using a subset of the cardboard and napkin validation data, such as the surfaces in Fig. 6.7(a,b), we formed two sequences of deforming meshes, textured them and projected them with known perspective camera to obtain noise-free images, as depicted in Fig. 6.7(c,d). We then added i.i.d. noise in the range $[-10, 10]$ to the image intensities. We reconstructed surfaces from a well-textured sequence and from a more uniform one. In Fig. 6.8 and 6.9, we plot reconstruction errors for both models without penalizing extension of the global mesh. It can be observed that the linear models perform poorly in the cardboard case. This can be explained by the fact that the mesh tends to stretch to explain the images, which gives a large error. This is less noticeable for the napkin sequence, since the deformations of a more flexible object tend to remain closer to a flat shape. In Fig. 6.10, we show similar results when using inextensibility constraints for both models. Note that the non-linear models are less affected by these constraints than the linear ones. This was expected

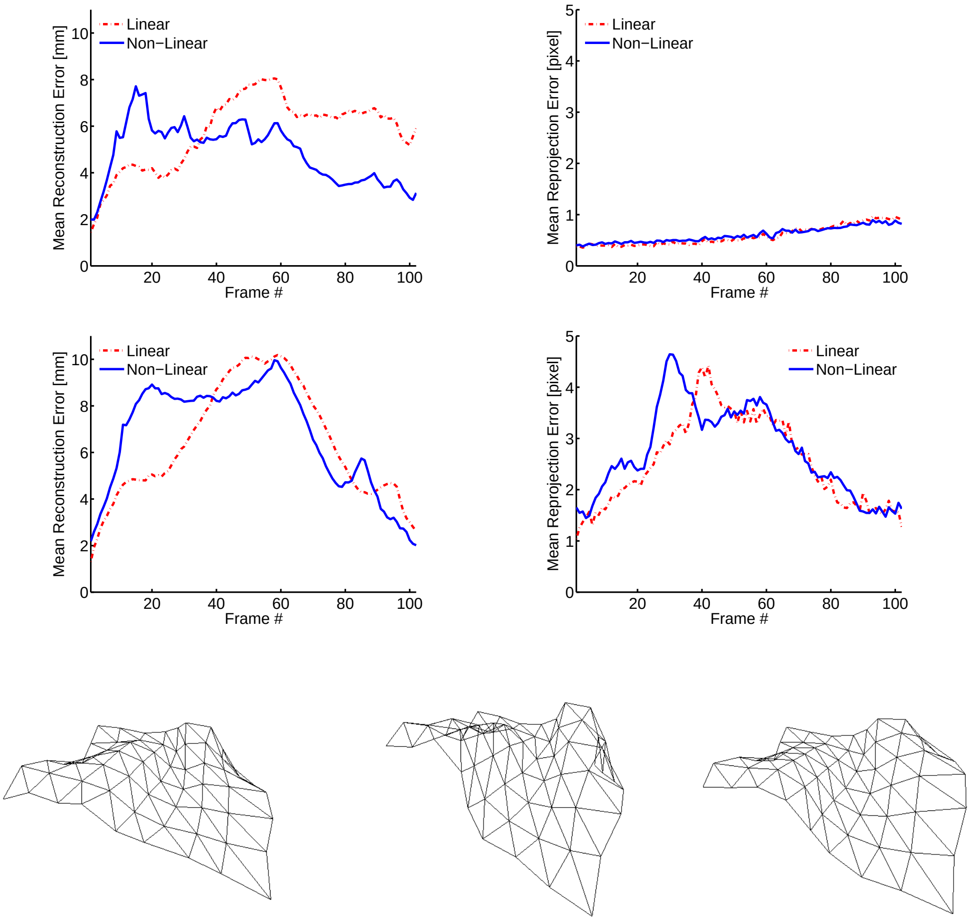


Figure 6.9: Same plots as in Fig. 6.8 for the napkin models. This time, the linear models perform as well as the non-linear ones. This can be explained by the fact that the deformations of a flexible material remain closer to the flat shape than that of a more rigid one. Thus, even when stretching, the linear models yield reasonable 3D errors, but visually less accurate shapes, as can be checked from the last row of the figure which depicts the linear (left) and non-linear (right) reconstructions of the same ground-truth (middle) corresponding to frame 60 of the less-textured sequence.

since the non-linear models give a better approximation of the density in shape space, and thus should implictly better satisfy the length constraints. In the case of poorly-textured cardboard, the non-linear results are even degraded with the introduction of inextensibility constraints. This can be explained by the fact that they are non-convex constraints, and can prevent the already complex objective function involved by the non-linear models to converge.

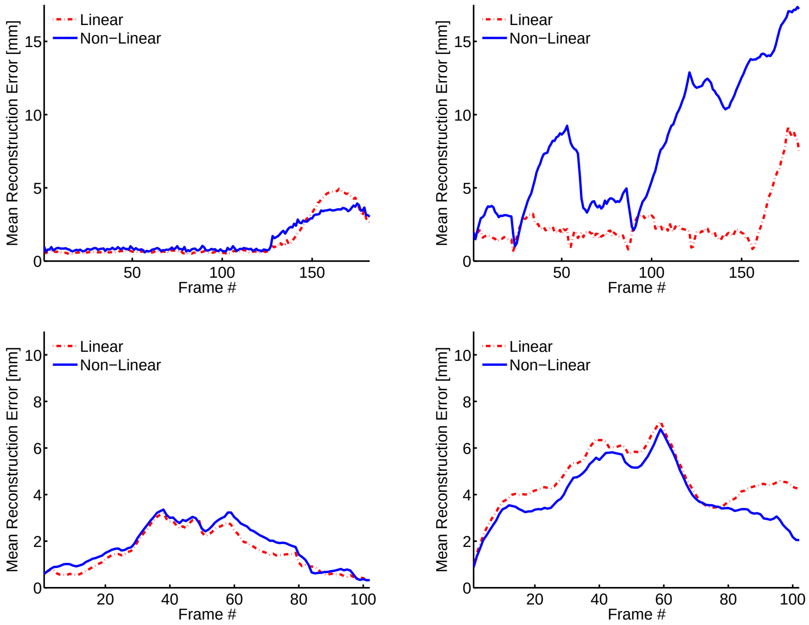


Figure 6.10: 3D reconstruction errors for the linear and non-linear models when using inextensibility constraints. Top: Mean vertex-to-vertex distance as a function of time for the cardboard textured (left) and less-textured (right) sequences. Bottom: Same plots for the napkin sequences. Note that in both cases, the linear models strongly benefit from the inextensibility constraints, whereas the non-linear ones are less affected.

6.4.3 Real Sequences

We then applied our local models to recover the shape of surfaces made of the same cardboard and paper tissue from real video sequences. Note that even in the cases where the global shape we track is rectangular, we had to assemble local models to represent it since the global shape is not the same as that of our training data.

We first applied our approach to the sheet of carboard of Fig. 6.11. The top row of Fig. 6.11 shows the behavior of our technique when there is absolutely no texture to anchor the surface. The recovered surface belongs to a family of equally-likely shapes whose vertices can slide across the surface, while their boudaries reproject correctly. Nothing in the image likelihood prevents this, since all facets look similar. Note that, without using shading cues, even a human eye could hardly differentiate between two such shapes. However, as shown in the second example of the figure, adding only very little texture disambiguates the reconstruction. Finally, when increasing only slighly the amount of texture, even more complex deformations can be recovered accurately, as shown in the third example. The results of this figure were all obtained with non-linear models.

We then tested our linear models on the same sheet of cardboard. With synthetic data, we

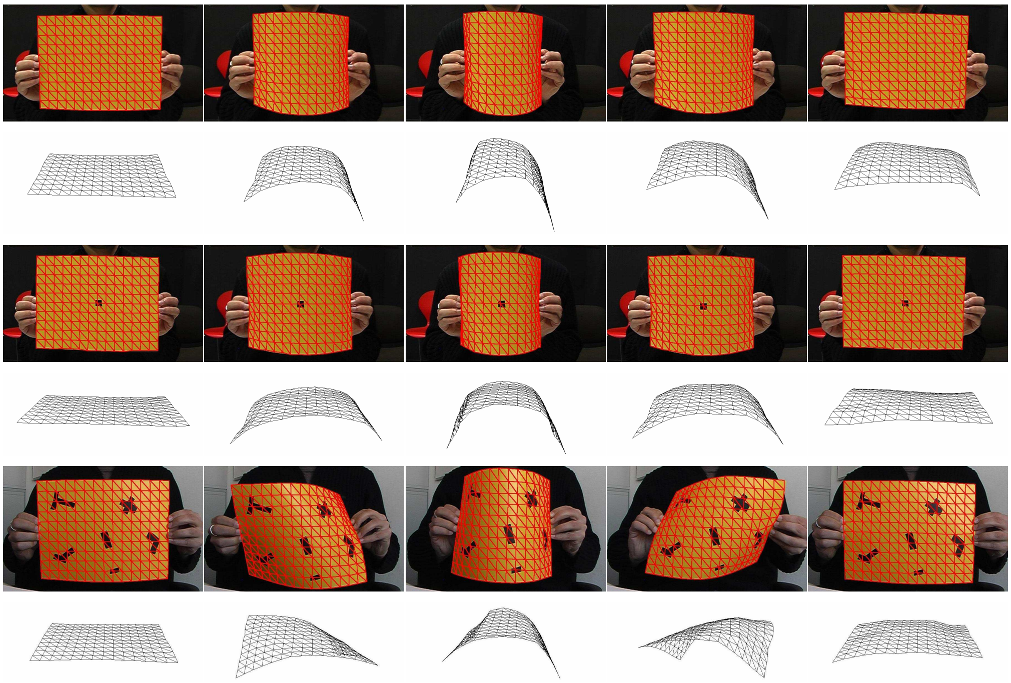


Figure 6.11: Reconstructing a rectangular piece of cardboard from a single video. In each of the three examples, we show the recovered surface overlaid in red on the original images, and the surface seen from a different viewpoint. As shown in the top rows, a complete absence of texture leads us to retrieve a surface that is plausible, but not necessary accurate. It is only one of a whole family of equally likely solutions. However, this problem is fixed by adding very little image information, as shown in the other two examples. We then recover deformations that match the real ones.

observed that the prior over the global mesh that enforces inextensibility was very helpful. To check if this was still the case in real sequences, we tracked the same sequence with and without those constraints. As can be checked in Fig. 6.12, inextensibility constraints truly improve the results.

One of the main advantages of local models over global ones is that they can be applied to represent very different shapes and topologies. We therefore assembled local cardboard models to form the circular shape of Fig. 6.13. Our models being made of rectangular patches, the mesh we use only roughly approximates the surface boundaries, which prevented us from using edge information. We nevertheless recover the correct 3D deformations. The second row of the figure shows results obtained with the non-linear models, and the bottom row, those computed with the linear ones. In this case, the linear models perform slightly better than the non-linear ones. However, this is to the price of additional inextensibility constraints. In the top row, we only show the reprojection of the non-linear

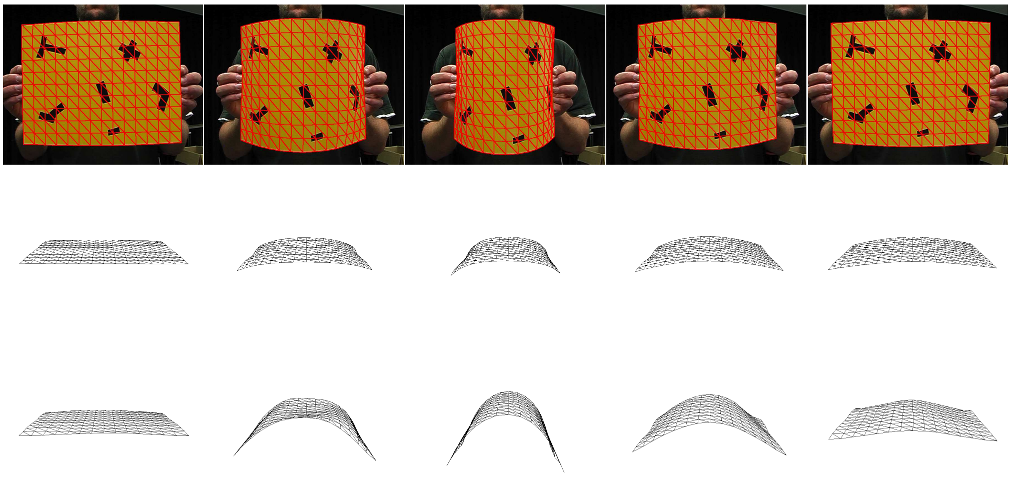


Figure 6.12: Reconstructing a rectangular piece of cardboard from a single video with linear local models. In the first row, we show the surface recovered with inextensibility constraints overlaid in red on the original images. In the middle, we show a side of the surface recovered without penalizing stretching. In the bottom row, we can see that inextensibility constraints improve the 3D shape, and therefore should be used in conjunction with the linear local models. Note that we do not show the reprojection of the compressed surface, since it yields similar images as in the first row.

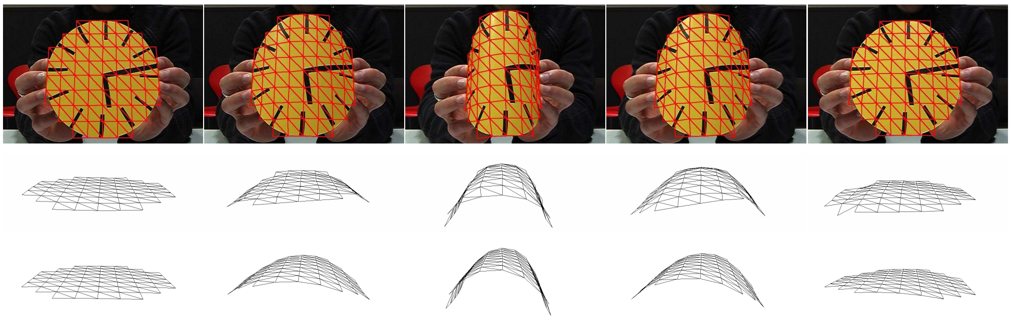


Figure 6.13: Reconstructing a circular piece of cardboard with the same local models. Note that assembling square patches only allows us to approximate the object's outline. This prevents us from using image edges, but does not stop us from successfully recovering the deformations. In the middle row, we show the results of the non-linear models, and in the bottom row, those of the linear ones.

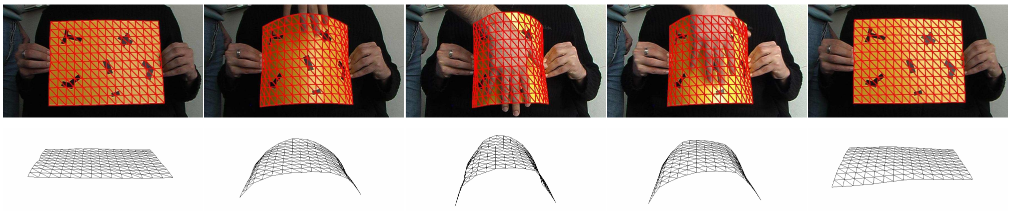


Figure 6.14: Despite a very large occlusion, we manage to reconstruct a deforming piece of cardboard in each frame of a sequence. Note that even if some small reconstrution errors occur, the global shape nevertheless matches the true one.

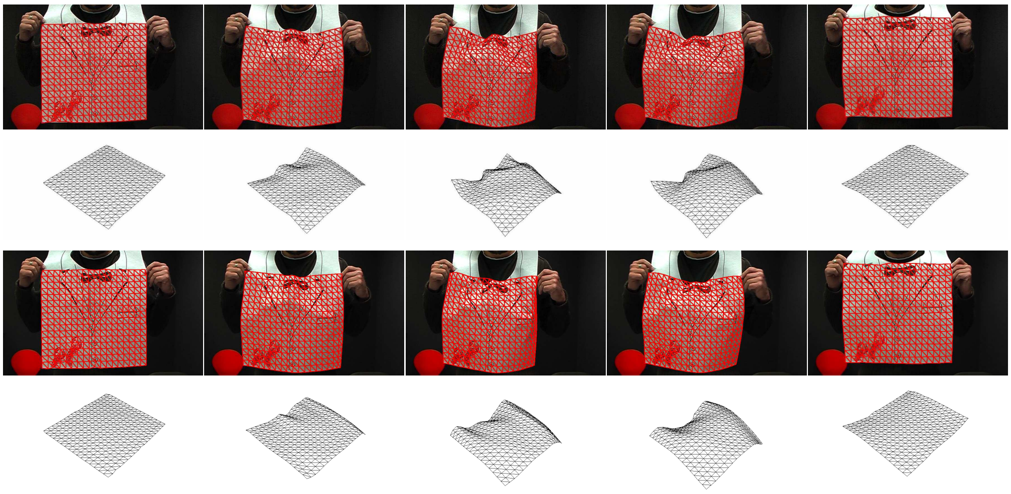


Figure 6.15: Reconstructing a much more flexible paper napkin. As opposed to cardboard, results obtained with non-linear models (first and second rows) are better than with linear ones (third and fourth rows). This confirms our intuition that complex deformations are non-linear.

results, since they are almost indistinguishable from that of the linear ones. Finally, in Fig. 6.14, we show that our models make our approach robust to occlusions.

We then applied our local models to recovering the shape of a much more flexible paper napkin. In Fig. 6.15, we show non-linear and linear results on the same sequence. This time, we show both reprojections, since the one of the results obtained with the linear models is noticeably worse than the other. This could be expected since it is well-known that highly flexible materials follow a non-linear behavior, which makes linear models not adapted. Furthermore, folds can appear between two vertices of the mesh, which makes inextensibility constraints inappropriate, and thus prevents the mesh from matching the image correctly. Nonetheless, thanks to our models, the results remain plausible. In Fig. 6.16, we show the results of our non-linear models on another sequence of the same napkin.

We applied our napkin models to recover the shape of a surface of different topology.

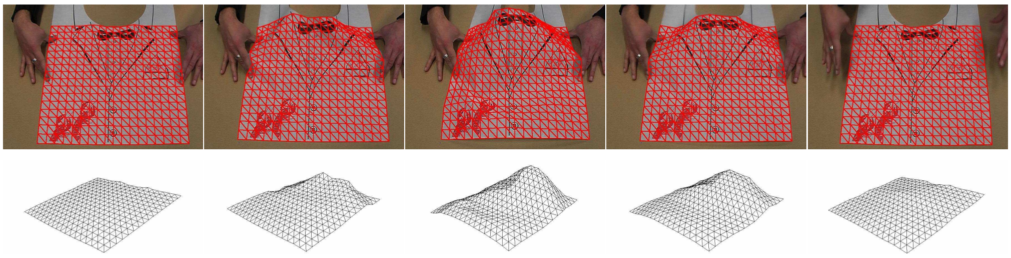


Figure 6.16: Reconstructing a different deformation of the same napkin. Even though there is little texture, the 3D shape of the surface is correctly recovered, as shown in the bottom row where the surface is seen from a different perspective.

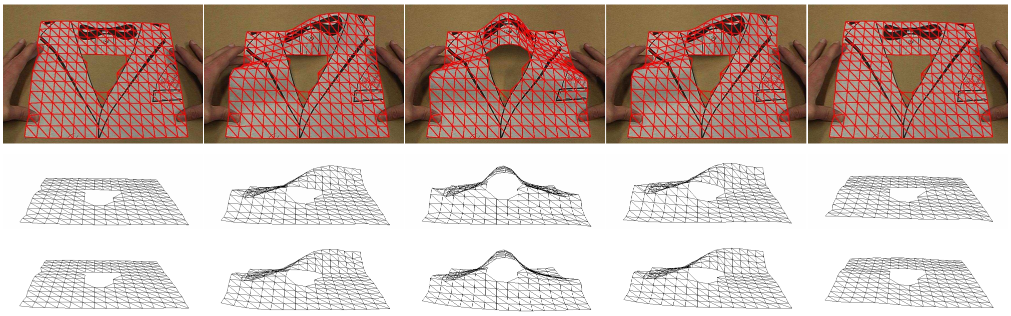


Figure 6.17: Reconstructing a napkin of different topology with non-linear (top) and linear (bottom) models. As with cardboard, assembling square patches only allows us to approximate the outline of the hole, but still lets us recover correct deformations.

Fig. 6.17 depicts the results of our non-linear and linear models on a video sequence of a napkin with a hole. As before, square patches cannot perfectly model the triangular hole, but still let us recover correct shapes. In this case, one might consider no explictly modeling the hole, and simply cover the whole rectangular surface with patches. However, as shown in Fig. 6.18, in some places, such as the upper boundary of the hole, the recovered shape does not perfectly reproject on the image anymore. This typically corresponds to the frames where the hole creates discontinuities in the global surface, thus modifying its behavior. Explicitly modeling the hole allows for such discontinuities.

Finally, since the linear local models do not enforce inextensibility, nothing prevents us from using them to track stretchable materials, as was demonstrated with our global models. Therefore, we applied our local models to the same sequence of a stretching balloon as in the global case. The results are shown in Fig. 6.19, and compared to the ones obtained with the global model. Note that the local ones seem to better match the texture of the balloon. However, they were obtained with template matching as opposed to correspondences, which can explain this improvement.

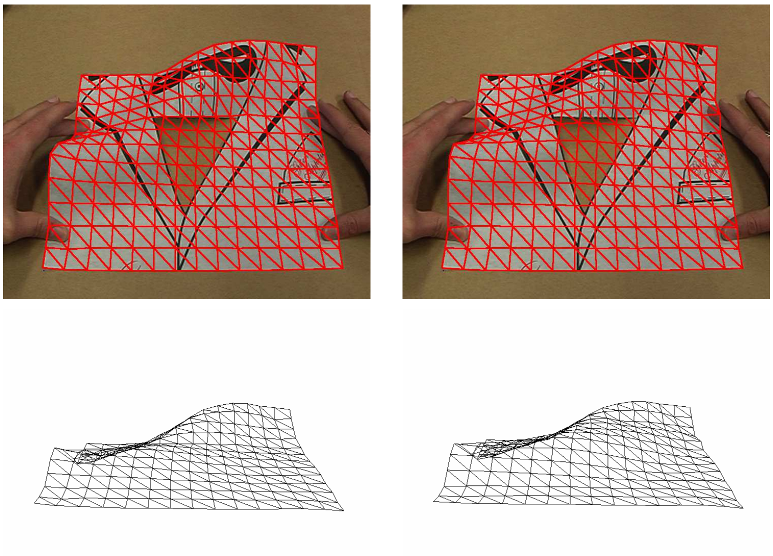


Figure 6.18: Modeling the napkin without explicitly accounting for the hole. The local models are replicated to cover the whole rectangular surface. Note that the surface does not always reproject correctly, as can be seen at the hole upper boundary. The hole creates discontinuities in the surface, which modifies the global behavior. It should therefore be modeled explicitly.

6.5 Closed-Form 3D Reconstruction

When there is enough texture to rely on feature points, we show that we can use the linear local models to perform 3D reconstruction in closed-form.

In Section 5.4, we showed that 3D reconstruction could be formulated as the solution to a linear system whose unknowns were the modes weights. The matrix of this system contained the correspondence equations as well as the regularization of these weights. Solving this system directly proved under-constrained. However, we showed that the remaining ambiguities could be overcome by solving a system of quadratic equations accounting for constant edge lengths, which can be done in closed-form using Extended Linearization [40]. Here, we show that the same formulation is still valid with our linear local models.

6.5.1 Constraining the Reconstruction via Linear Local Models

In Sections 3.2 and 5.4, we formulated shape recovery as the solution to a linear system of equations. Assuming that we have more equations than unknowns, which is true in practice, we obtain a solution in the least-squares sense. If we observe the first term of the negative log likelihood of Eq. 6.5, we notice that, when using the mean prediction of the linear formulation given by Eq. 6.16, it becomes a sum of squares. Therefore, for each

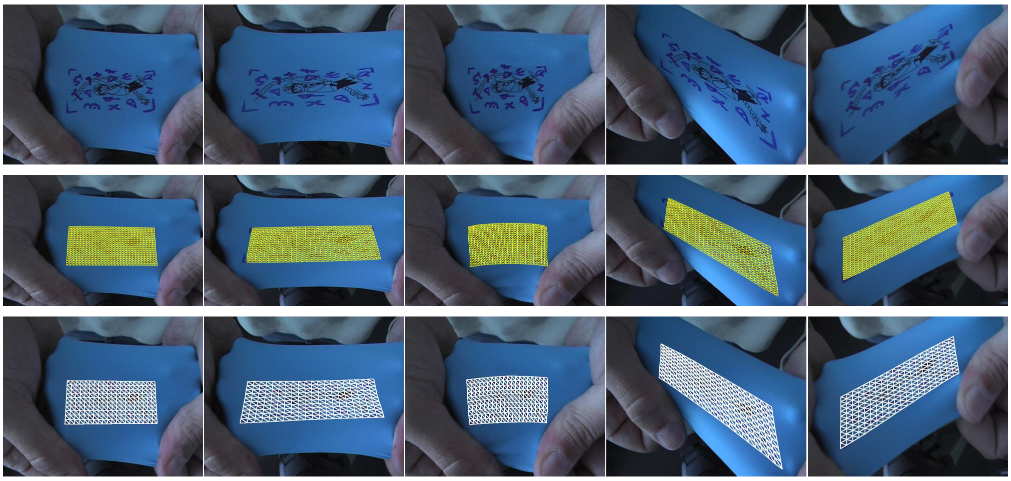


Figure 6.19: Using local models to track the same extensible surface as in the global case. In the top row, we show the original images, the second row displays our results with a global model, and in the bottom row, we show the surfaces obtained with local models. Note that the improvement in texture matching most probably comes from using template matching rather than correspondences. Furthermore, the change of mesh resolution was only introduced for convenience of use with the local models.

patch $\mathbf{y}'$ of a surface, we can re-write this term as the least-squares solution to

$$\begin{aligned} \mathbf{y}' &= \mu_l(\mathbf{y}') \\ &= \mathbf{y}_0 + \mathbf{S}\mathbf{S}^T(\mathbf{y}' - \mathbf{y}_0) \ . \end{aligned} \tag{6.27}$$

Similarly, the prior over the latent variables that prevents them from becoming overly large can be re-written as solving in the least-squares sense

$$\Lambda^{-\frac{1}{2}}\mathbf{S}^T(\mathbf{y}' - \mathbf{y}_0) = 0 \ . \tag{6.28}$$

Note that this formulation differs from the original term $\|\mathbf{x}'\|^2$ of Eq. 6.5 by a factor $\Lambda^{-1/2}$. This is due to the fact that our modes are normalized, by contrast with the standard probabilistic PCA formulation where the norms of the modes are proportional to the values on the diagonal of Λ. This formulation follows that of the regularization in the previous chapter and forces the modes weights to conform to the distribution of the training data.

This, in conjunction with correspondence equations, lets us write the solution to 3D reconstruction under the linear local models as

$$\begin{bmatrix} \sigma_C^{-2}\mathbf{M} & \mathbf{0} \\ \tilde{\mathbf{T}} & -\tilde{\mathbf{T}}\tilde{\mathbf{y}}_0 \\ \tilde{\mathbf{S}} & -\tilde{\mathbf{S}}\tilde{\mathbf{y}}_0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{y}}' \\ 1 \end{bmatrix} = \mathbf{0} \ , \tag{6.29}$$

where $\tilde{\mathbf{T}}$ is the matrix containing $(\mathbf{I} - \mathbf{S}\mathbf{S}^T)$ for the different patches of the global surface, thus encoding Eq. 6.27. Similarly, $\tilde{\mathbf{S}}$ contains the $\Lambda^{-1/2}\mathbf{S}^T$ term for each patch to account for Eq. 6.28. As in Eq. 6.19, the lines corresponding to the error between patch shape and mean prediction can be weighted according to the number of patches each vertex belongs to. Finally, σ_C is a constant set to the variance of the expected reprojection error.

Solving this system yields a shape that reprojects correctly on the image, and simultaneously remains close to the linear local models prediction. However, as with global models, the linear local models do not yield a well-posed problem. The corresponding linear system still has a number of small eigenvalues, since inextensibility constraints are not enforced.

6.5.2 Inextensibility Constraints

To enforce inextensibility of the mesh, we can follow the same idea as in Section 5.4.2. We first express the solution of Eq. 6.29 as a linear combination of the system's N_l eigenvectors $\mathbf{l}_i$ associated to its smallest eigenvalues. Then, we seek to recover the weights β_i of the combination that satisfy the inextensibility constraints.

As before, these constraints can be written as quadratic equations, which we can linearize by introducing new unknowns for the quadratic terms. This results in a system of the form

$$\hat{\mathbf{D}}\hat{\mathbf{b}} = \hat{\mathbf{d}} \ , \tag{6.30}$$

with $\hat{\mathbf{b}} = [\beta_1, \cdots, \beta_{N_l}, \beta_1\beta_1, \cdots, \beta_1\beta_{N_l}, \beta_2\beta_2, \cdots, \beta_{N_l}\beta_{N_l}]^T$. As with global models, $\hat{\mathbf{b}}$ contains the linear terms, since the system also encodes the additionnal linear equation that constrains the last coordinates of $\beta_i\mathbf{l}_i$ to sum up to 1. However, since our unknowns are the vertices coordinates and not the modes weights anymore, no equation of this system links the linear and the quadratic terms together. Therefore, we could obtain solutions where the relationships between these terms are violated.

Fortunately, this can again be solved by using Extended Linearization [40]. However, in this case, as opposed to the global model approach, we simply want to link the linear and quadratic terms together. Therefore, we only multiply the last linear equation by the linear monomials, thus creating N_l additional equations of the form

$$\sum_{j=1}^{N_l} \beta_i\beta_j\mathbf{l}_{j,3N_v+1} = \beta_i \ , \tag{6.31}$$

without the need of additional unknowns. This, in practice, proved sufficient to enforce the constraints between linear and quadratic terms and to obtain a good recovered shape. Nonetheless, nothing prevents us from using Extended Linearization more extensively as in the global case. The correct number N_l of eigenvectors to use was chosen in the same fashion as for the global case, by testing several values and choosing the one that, for a small enough reprojection error, gives the smallest mean edge length variation.

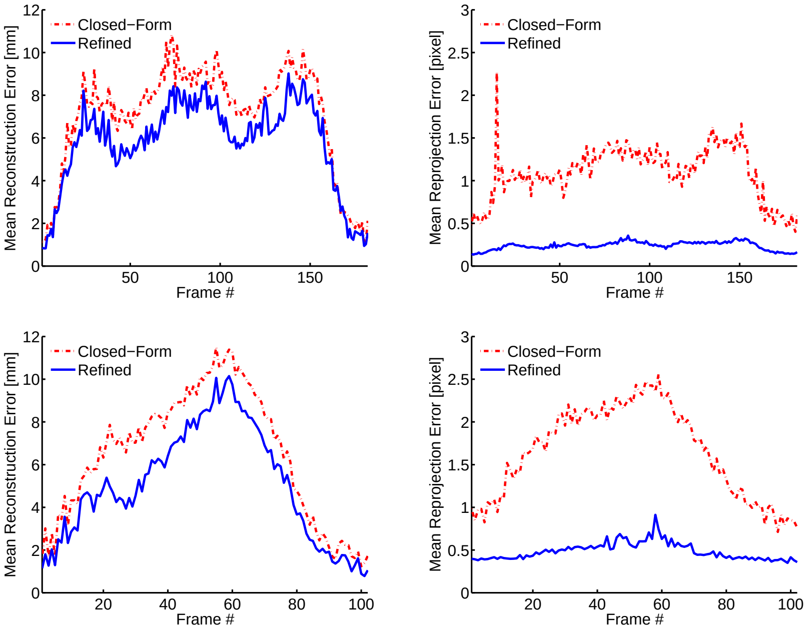


Figure 6.20: Mean vertex-to-vertex distance (left) and mean reprojection error (right) for the closed-form reconstruction using linear local models. The dashed red curve corresponds to the closed-form solution, and the solid blue one to the refined solution after optimization. Top row: The errors were computed for the textured cardboard sequence from correspondences synthetically sampled on the facets of the mesh. Bottom row: Same plots for the textured paper napkin sequence.

6.6 Experimental Results

We tested our closed-form reconstruction on synthetic data as well as real sequences. Note that we rely on image correspondences only. This forced us to apply our technique to more textured objects than when tracking.

6.6.1 Synthetic Data

We used the same synthetic sequences as in the tracking case to quantitatively evaluate our method. However, rather than using image noise, we sampled the barycentric coordinates of the facets of the ground-truth meshes to obtain 3D-to-2D correspondences to which we added zero-mean gaussian noise with variance 5. Fig. 6.20 depicts the same 3D reconstruction and reprojection errors as for tracking when using all the modes. We further refined the closed-form solution by using it as an initialization to the tracking algorithm presented in Section 6.3. The dashed red curve corresponds to the closed-form solution, and the solid blue one to the refined solution. Note that the closed-form solution error is close to that ob-

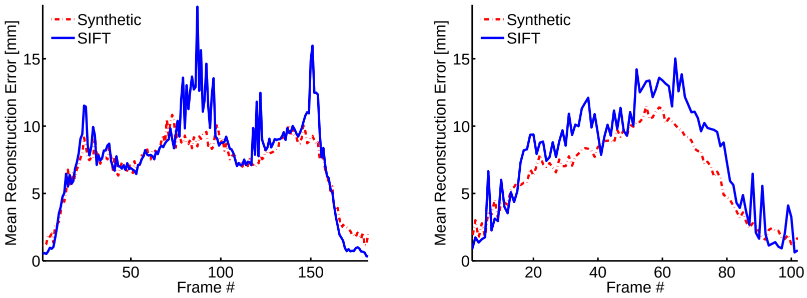


Figure 6.21: Similar plots as in Fig. 6.20, but with more realistic matches obtained with SIFT [103]. Left: We plot the errors obtained with synthetic matches (dashed red) and with SIFT matches (solid blue) as a function of time for the cardboard case. Right: Same plots for the paper napkin sequence.

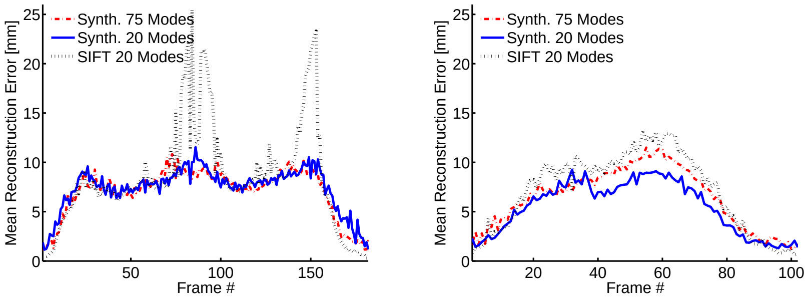


Figure 6.22: We studied the influence of the number of modes by computing the solutions with 20 modes rather than the original 75 ones. Left: We show the errors obtained from synthetic matches with 75 modes (dashed red) and 20 modes (solid blue), and from SIFT matches with 20 modes (dotted black) for the cardboard case. Right: Same plots for the paper napkin sequence.

tained by tracking the surface, and that refining the solution does not dramatically improve the results. Since our matches are synthetic, there is no need to test with different textures and we used only the well-textured images.

To study the more realistic case where we cannot establish correspondences synthetically, we used SIFT [103] to find matches in noisy textured images. Since SIFT can give outliers, we iterated five times our closed-form solution with a decreasing weight for the regularization of the latent variables, and re-weighted the correspondences based on their reprojection error. In Fig. 6.21, we compare the 3D reconstruction errors obtained with synthetic matches and SIFT matches. As can be checked from the plots, the solution with SIFT matches remains close to the synthetic one, except in occasional cases that either presented too many outliers, or where parts of the surface were lacking correspondences.

We also studied the influence of the number of modes. Rather than taking all 75 modes

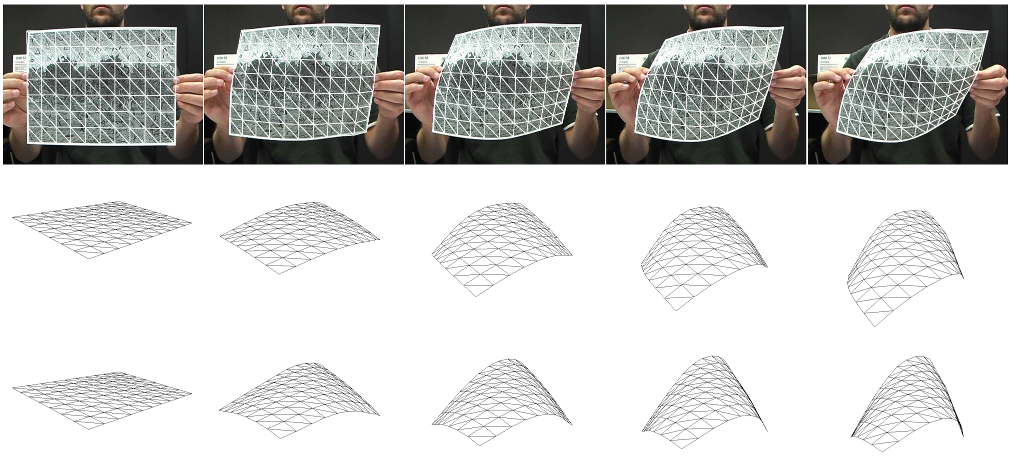


Figure 6.23: Recovering the shape of a piece of paper. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

as in the previous cases, we computed our results with only 20. The errors are depicted in Fig. 6.22, where we plot the results obtained from synthetic matches with all modes in dashed red, those obtained from synthetic matches with 20 modes in solid blue, and those obtained from SIFT matches with 20 modes in dotted black. Note that the number of modes has little influence on the quality of the results.

6.6.2 Real Images

Finally, we applied our closed-form solution to recover the shape of different surfaces from real images. For all the following cases, we used dense correspondences obtained as described in Section 3.3.1.2. Furthermore, since we deal with other materials than the cardboard and paper napkin of the previous sequences, we computed the deformation modes synthetically, as described in Section 5.1.3 but for small patches of 5×5 vertices, and used all of the modes for reconstruction. As before, even though the surfaces recovered here are rectangular, they were modeled as combinations of linear local representations. Additionally, despite the fact that we used video sequences, the images are treated independently and nothing links our results from one frame to the next.

As a first case, we recovered the deformations of the piece of paper of Fig. 6.23. In the first row of the figure, we reprojected our closed-form solution on top of the original images. In the middle row, we show a side view of our closed-form solution. As with global models in Section 5.4, we used this solution as an initialization for a simple Gauss-Newton optimization scheme. We display the refined solution after 5 optimization steps in the bottom row. Fig. 6.24 depicts similar results for a more complex deformation of the same piece of paper.

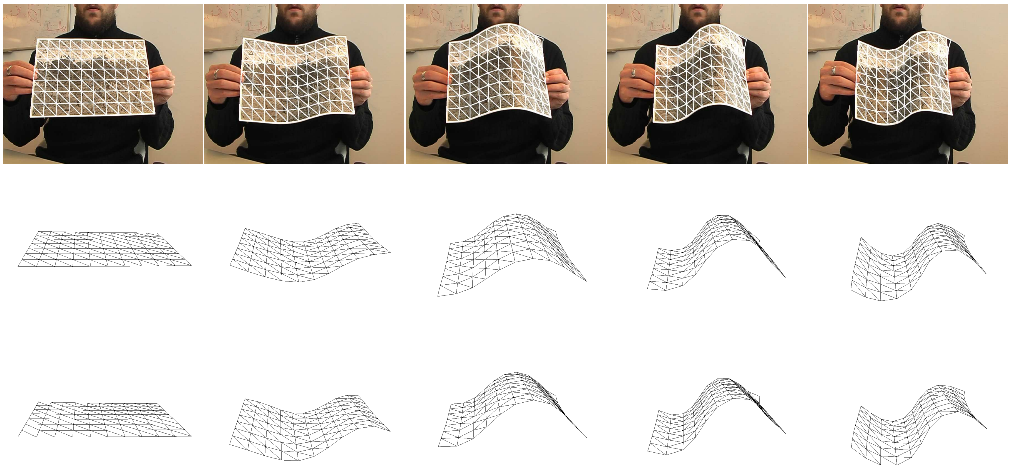


Figure 6.24: Recovering more complex deformations of a piece of paper. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

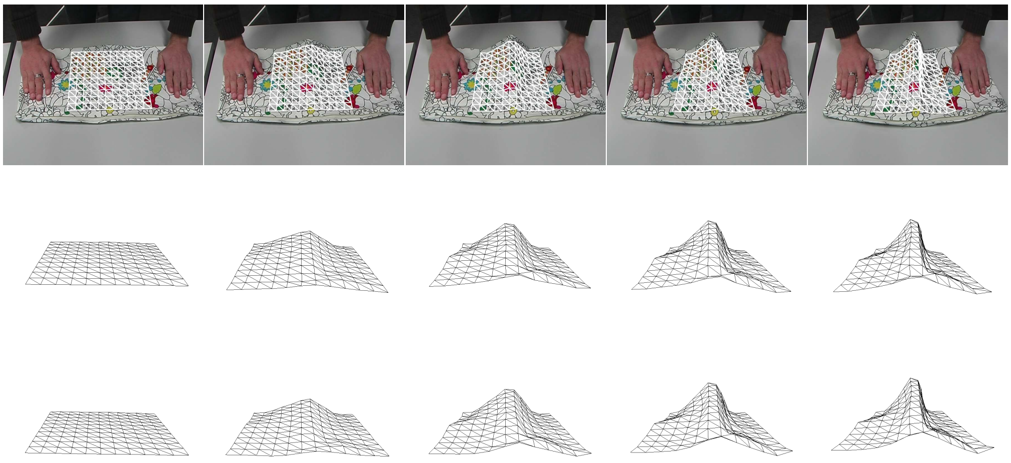


Figure 6.25: Reconstructing a sharp fold in a piece of cloth. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

Since by using synthetic modes, nothing prevents us from reconstructing different materials, we applied our technique to recover the deformations of a piece of cloth and of a plastic bag. Figs. 6.25 and 6.26 depict our results on thoses cases. Note that our local models manage to reconstruct sharp folds and very local deformations.

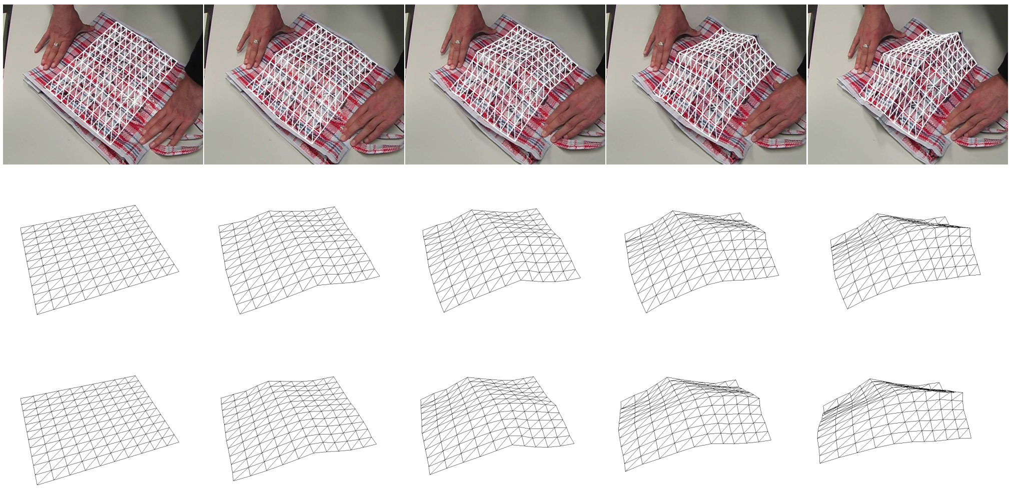


Figure 6.26: Recovering the shape of a plastic bag. From top to bottom: Mesh computed in closed-form overlaid on the input image, side view of that mesh, refined mesh after 5 Gauss-Newton iterations.

6.7 Conclusion

In this chapter, we have introduced local deformation models to recover the shape of deformable surfaces from a single viewpoint. The main advantages of local models over global ones are that they can be learned from smaller amounts of training data, since small patches of a surface deform less than the surface itself, and that they can be assembled to represent any surface shape and topology using a PoE formalism. Our local models are formulated as Gaussian Process Latent Variable Models, which let us define linear or non-linear mappings from a low-dimensional manifold to the high-dimensional shape space. They can be used both to track deformations from image to image and, when there is enough texture, to recover the shape of a surface from individual images.

7 Comparative Results

In this chapter, we compare the different models that we have presented in Chapters 4, 5, and 6, as well as other state-of-the-art techniques. First, we compare the deformation spaces covered by our various learned models. We then evaluate the accuracy of the models on synthetic and real data. Since the different models sometimes work with different kinds of inputs, we chose the image sequences accordingly, and, in some cases, only evaluate some of the models.

7.1 Competing Techniques

As discussed in Chapter 2, many methods have been investigated over the years to constrain 3D shape recovery. Since a majority rely on physics-inspired constraints, for comparison purposes, we will consider two physics-based approaches to regularization: imposing smoothness either by minimizing a simple quadratic term or by performing modal analysis on a more sophisticated stiffness matrix.

7.1.1 Smoothness Assumptions

A very popular approach to constraining 3D reconstruction is to assume that the surface is locally smooth. This was a key components of Active Contours [84] and has often been used for stereovision, or non-rigid 2D registration [127]. In regular 3D meshes, this amounts to minimizing the sum of the squared curvatures of the surface, which can be written as

$$E_D = \sum_{\{i-1,i,i+1\}} (2x_i - x_{i-1} - x_{i+1})^2 + (2y_i - y_{i-1} - y_{i+1})^2 + (2z_i - z_{i-1} - z_{i+1})^2 \ , \quad (7.1)$$

where $\{i-1, i, i+1\}$ spans the set of all triplets of aligned vertices. When reconstructing a surface from images, we minimize this term together with the usual image term. Whereas the 2D version of these constraints is perfectly suitable, it becomes unnatural in 3D since we expect one direction to deform more than the others. As will be shown below, this indeed limits the applicability of this method.

7.1.2 Physics-based Deformation Modes

The physics-based approach has been very popular in Computer Vision. Since 3D reconstruction often involves high-dimensional models, modal analysis [122, 43] was often

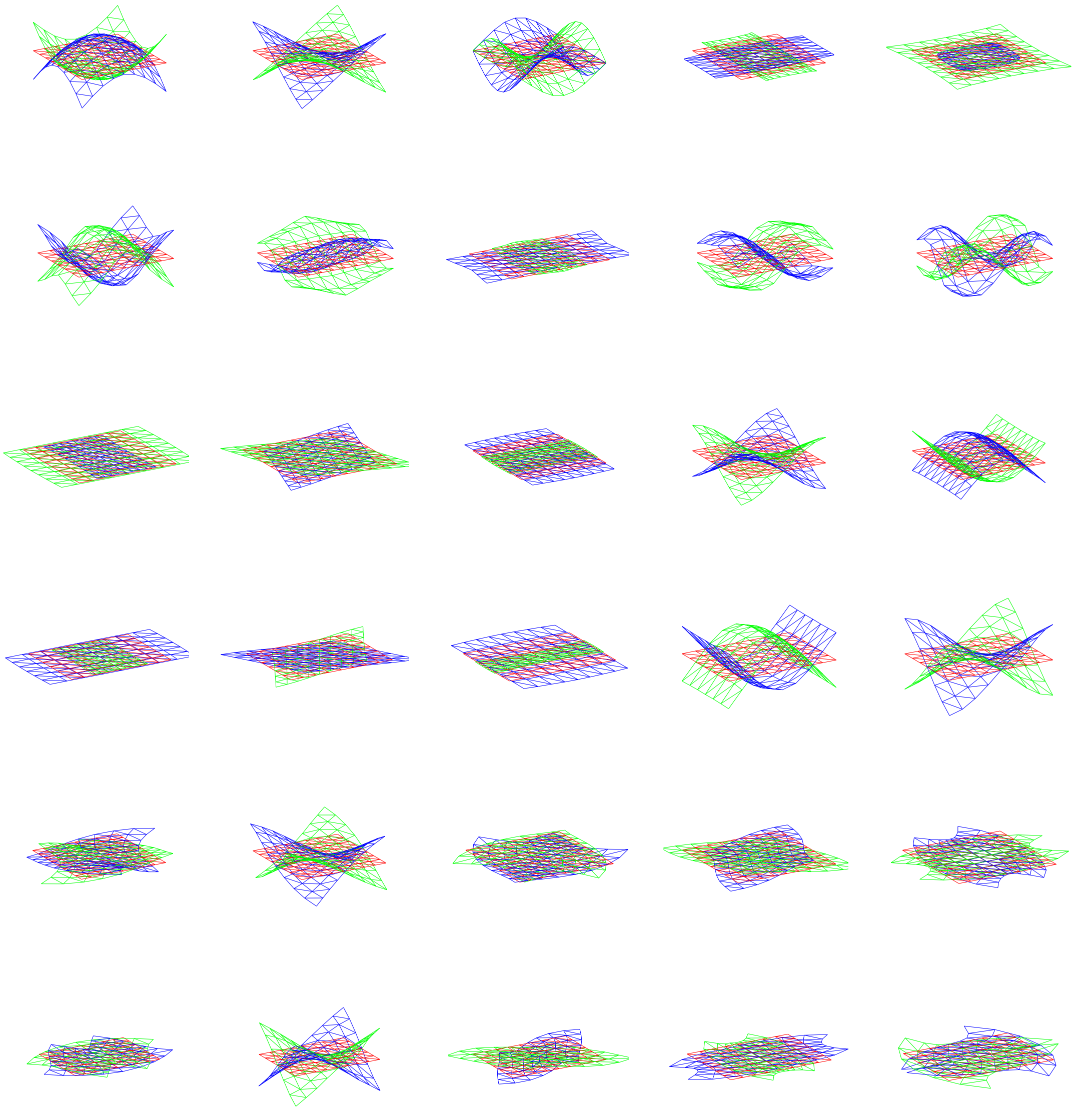


Figure 7.1: Effect of adding (green) or subtracting (blue) a single deformation mode to the rest shape (red). From top to bottom, modes were obtained with real cardboard data, real napkin data, synthetically generated meshes of two different topologies, a stiffness matrix for cardboard, and a stiffness matrix for the napkin. The plots display the first five modes other than global motion.

applied to reduce the number of parameters by describing a deformation as a linear combination of vibration modes. This follows the same idea as our linear models. However, in

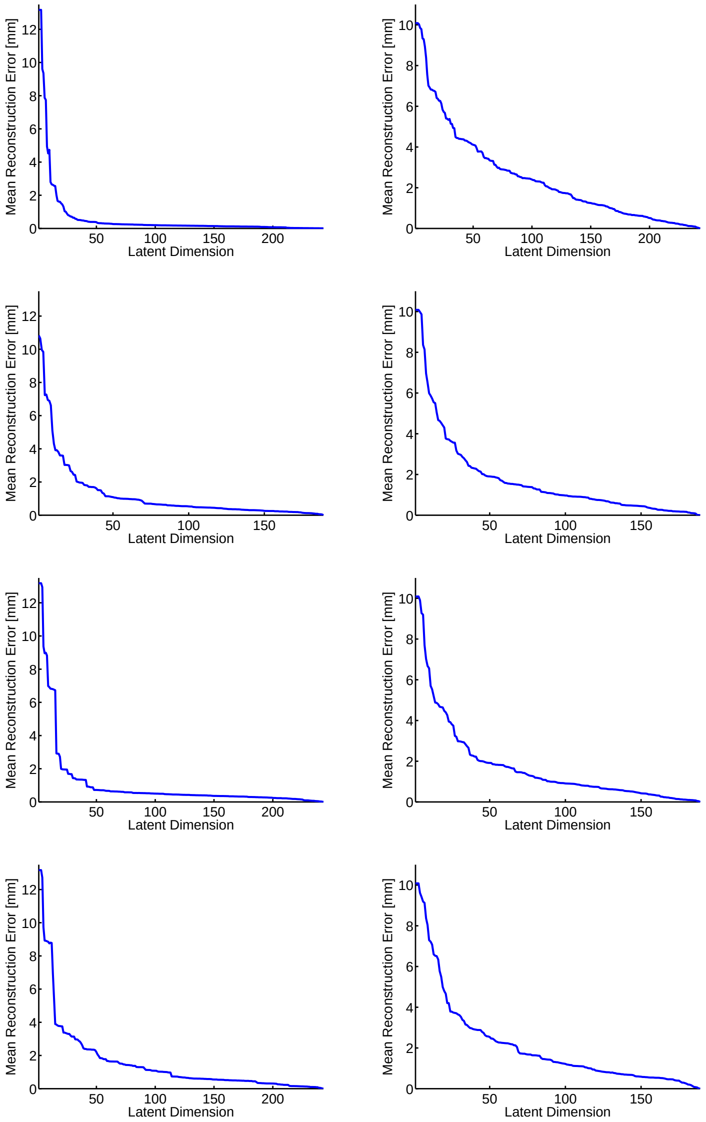


Figure 7.2: Comparison of the reconstruction error as a function of the number of modes for the different linear global models. On the left, the error was computed on the cardboard validation data, whereas on the right it was done for the napkin case. From top to bottom, we display such errors for the cardboard model, the napkin model, the model obtained from synthetic data, and the physics-based models.

this case, the modes are obtained by eigen-decomposition of a stiffness matrix rather than from training data.

Following this approach, we built such a stiffness matrix for thin shell structures [92],

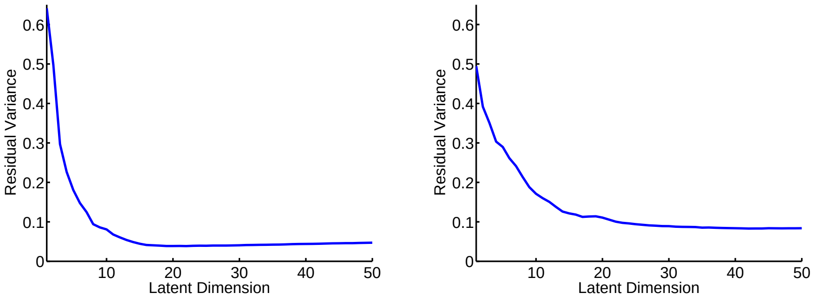


Figure 7.3: Residual variance obtained with Isomap on the cardboard (left) and napkin (right) global training data. These plots suggest latent dimensions 16 for the cardboard and 30 for the napkin.

which typically correspond to our objects of interest. We then replaced our PCA modes by the eigenvectors of that stiffness matrix and performed shape recovery in the same manner as described in the previous chapters.

Another possibility to apply physics-based methods could be to create training data. This would involve randomly applying forces to a finite element model, and simulate deformations. However, solving such a problem is known to be very complex, especially in the case of large deformations of thin structures.

7.2 Comparing Deformation Spaces

In this section, we compare our learned models by measuring their reconstruction errors on test data, as well as by visually displaying the effect of some of their latent variables. For the linear local and global models, we can compute the deformation modes by simulating training data using the angle-based parameterization proposed in Section 5.1.2, by using real training data as explained in Section 6.1.4, or through modal analysis as explained above.

7.2.1 Global Models

First, we study the global modes obtained with the three different techniques mentioned above. In Fig. 7.1, we visually compare the first five deformation modes that do not involve a global motion of the surface. In the first row, we show the modes obtained from the real cardboard data and in the second one, those obtained with the napkin data. The third and fourth rows depict the modes obtained from synthetic angle-based data with different topologies, and the fifth and sixth ones, those corresponding to the physics-based models for the cardboard and for the napkin, respectively. The two different physics-based models were obtained by using values of Young’s modulus, Poisson’s ratio and thickness that approximately correspond to the two materials.

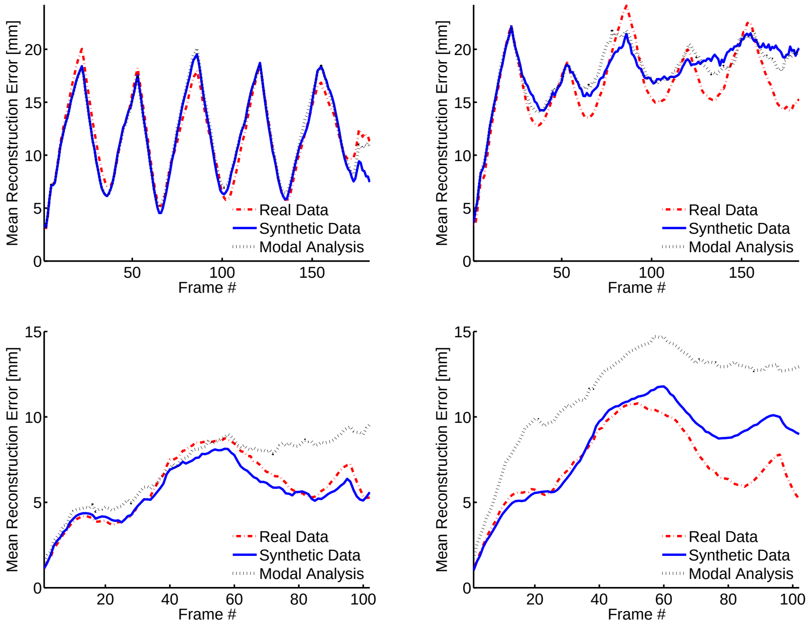


Figure 7.4: Reconstruction errors on synthetic images generated from validation data. We applied a nicely textured image (left) or a much more uniform one (right) to ground-truth meshes, and projected them with known camera. The top row corresponds to a piece of cardboard, and the bottom one to a paper napkin. In each plot, we show the results for the global models obtained from real data (red), synthetic data (blue), and modal analysis (black). Note that the latter performs significantly worse as the others in the napkin case.

As can be seen from the figure, using different parameter values in the physics-based models produces similar modes, but in a different order. Furthermore, these modes are quite different from the ones obtained from real and synthetic data. Similarly, the modes obtained with cardboard data are slightly different from those of the paper napkin, but are closer to the synthetic ones. As could be expected, changing the topology of the mesh does not influence the first few synthetic modes.

We then validated these different models in a similar manner as in Section 6.4.1, by computing their mean reconstruction error for various test shapes. To do so, we used the global test shapes that were kept aside during motion capture of the cardboard and napkin surfaces. As for the local models, we computed the mean prediction given by Eq. 6.16 and took the error as the average vertex-to-vertex distance between such prediction and the true test shape. Fig 7.2 shows such distances as a function of the number of modes for the cardboard data on the left, and for the napkin on the right. Note that, since the global napkin and cardboard data do not have the same size, we had to cut our cardboard examples to train a new model with the same shape as the napkin one. It can be observed

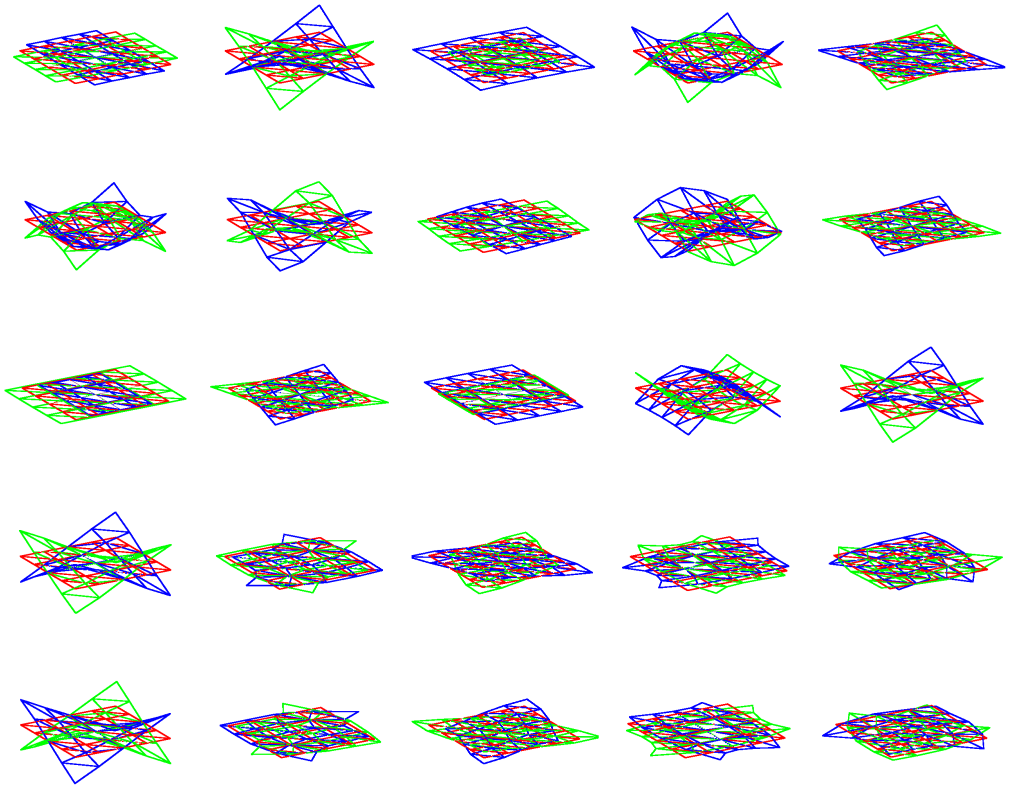


Figure 7.5: Visual interpretation of the local deformation modes. From top to bottom, we show the cardboard, napkin, synthetic, cardboard stiffness, and napkin stiffness modes. Note that they closely ressemble the global ones.

that, whereas the models learned from real data yield the curves that decrease fastest on the test set for the same material, this is not always the case for another material. By contrast, the models learned from synthetic data perform relatively well on both test sets, though slightly worse than the real models. This suggests that, when no real data is available, using synthetically generated ones still yields good precision.

Even in the global formulation, nothing in theory prevents us from learning a non-linear model as was done for the local representations. Since training a non-linear model is computationally much more expensive, we applied Isomap to find the correct latent dimension, to avoid having to learn a model in 50 different dimensions. The resulting residual variances are shown in Fig. 7.3, and suggest dimensions 16 for the cardboard and 30 for the napkin. Given these latent dimensions, we then tried to learn non-linear GPLVM's as with patches data. Unfortunately, even though we tried several numbers of inducing variables, learning never converged. Thinking this might have been an artifact of sparsification, we tried training a full model. However, Matlab ran out of memory. This again strongly con-

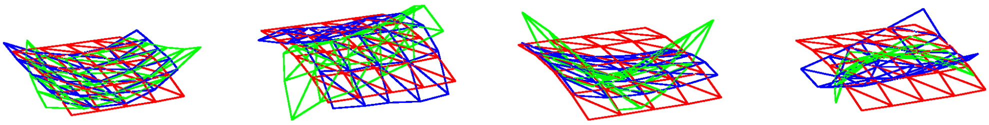


Figure 7.6: We computed the mean predicitions of the non-linear local models for the minimum (green) and maximum (blue) values for a single latent dimension while setting the others to zero. Here, we show the deformations along the 4 possible dimensions for our cardboard model.

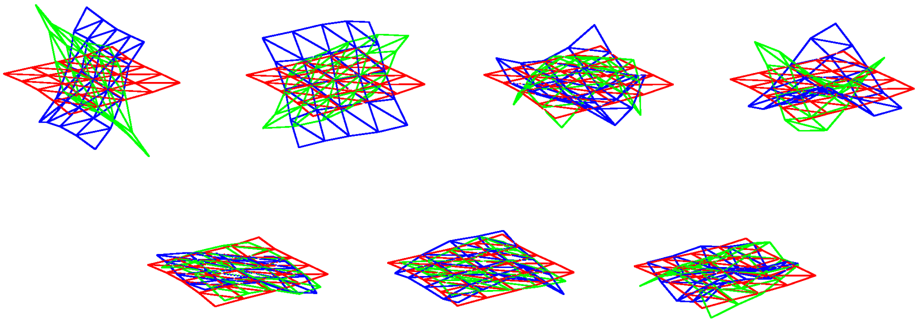


Figure 7.7: Same plot as in Fig. 7.6, but for the 7 dimensions of the paper napkin model.

firms the superiority of local models over global ones.

Finally, we compared the accuracy of the different models when using them to recover the shape of a surface from images. To do so, we applied our method on the same synthetic images that we used in Section 6.4.2, which were obtained from the cardboard and napkin validation sets and are depicted by Fig. 6.7. At each frame, we computed the mean vertex-to-vertex distance between our reconstruction and the ground-truth mesh. In Fig. 7.4, we show these errors for the textured and more uniform cardboard and napkin sequences. In the carboard sequence, all models perform roughly equivalently. However, in the napkin case, the model learned from real data performs best, closely followed by the synthetic one. Modal analysis has much more troubles recovering the correct shapes. This is probably due to the fact that the chosen material parameters were not exactly correct. This sensitivity to parameters reflects the weakness of such representations.

7.2.2 Local Models

We then performed similar comparisons in the case of local models. As for global ones, we plot the influence of the first five non-rigid modes of the linear models. As can be seen in Fig. 7.5, even with 5×5 patches, the various approaches yield different modes.

Additionally, we studied the effect of the different dimensions of the non-linear models. To do so, we computed the mean predictions given by the models by varying a single latent dimension at a time between its minimum and maximum values, and setting the others to zero. Note that if the low-dimensional space is not dense, this might not always be meaningful. Furthermore, since we applied a non-linear technique, a single direction can represent more complex deformations than in the linear case. The resulting mean predictions are depicted in Figs. 7.6 and 7.7 for the cardboard and napkin, respectively.

As before, we also compared the reconstruction error for increasing latent dimensions. We computed the same average mean vertex-to-vertex distances between the model mean prediction and the true shape as in Section 6.4.1. Having shown in the previous chapter that the non-linear models perform better in much smaller dimensions than the linear ones, here we only compare the different linear models. As can be checked in Fig. 7.8, the linear models trained with real data perform again better on their corresponding test sets, and, as with global models, the angle-based parameterization yields modes that perform relatively well independentely of the material.

Since the training examples obtained by randomly varying the determining angles of the mesh yield good and stable linear models, we tried to use them as training data for a non-linear GPLVM. We applied Isomap to find the correct latent dimension. Even though the error in Fig. 7.9(a) is already very low for dimension 4, it really stabilizes at dimension 10, as shown by the plot in Fig. 7.9(b). In theory, as explained in Section 5.1.2, this number should be 16, without accounting for translations that were removed. The fact that the determining angles are bounded, and that some of them only yield very small deformations can explain the difference.

Since we aim at recovering smoother cardboard or napkin data, we can expect the true dimension to be lower than that. We therefore learned models for dimensions between 1 and 10, and validated them on either the coardboard or the napkin data. The resulting reconstruction errors are shown in Fig. 7.9(c). Note that these models perform poorly compared to the ones trained on real data. Furthermore, using sparsification techniques prevented the training to converge. We therefore had to learn full GPLVM's, which is computationally expensive and would make our tracking algorithms excessively slow.

As with global models, we compared the reconstruction accuracy of the different local models on synthetic images. Fig. 7.10 shows the reconstruction errors on the same sequences as in the global case, but with local models. We only plot results obtained with the different linear models, since a comparison with non-linear ones was shown in the previous chapter. As can be checked from the figure, the models learned from real data perform best, especially with poorly-textured surfaces. As with global models, modal analysis gives the worst results on the napkin data.

7.3 Comparison of our Shape Recovery Techniques

We now compare the performances of the different methods we proposed to recover the shape of surfaces from images. Since some methods rely more strongly on texture, or

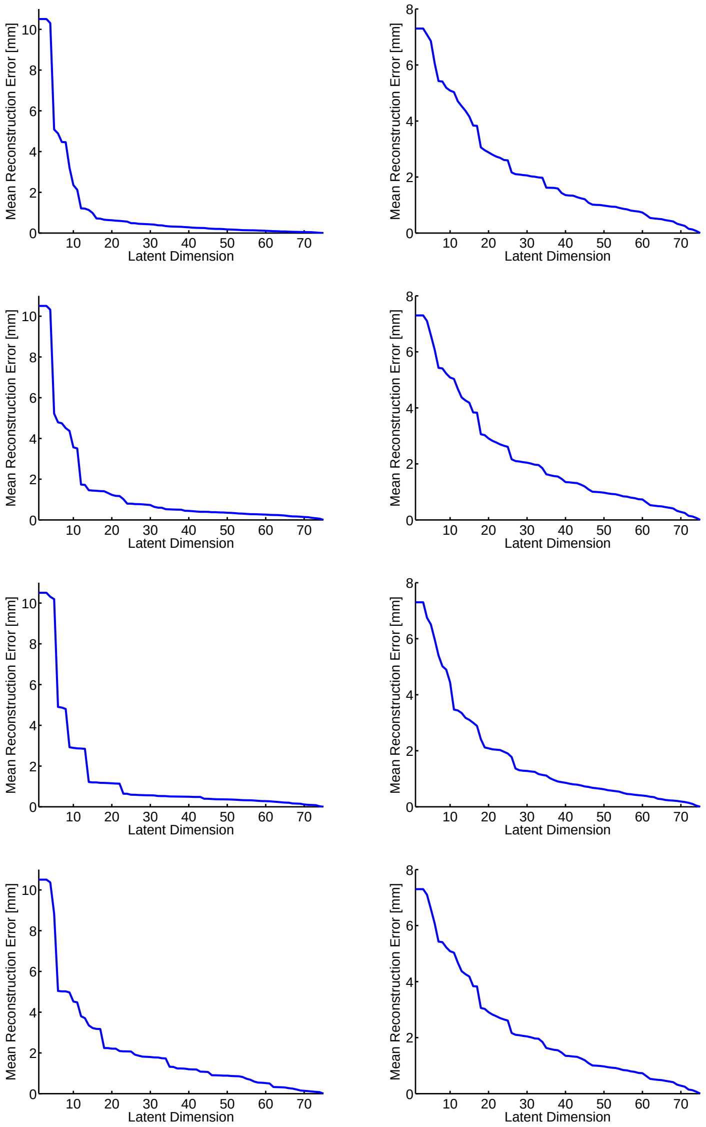
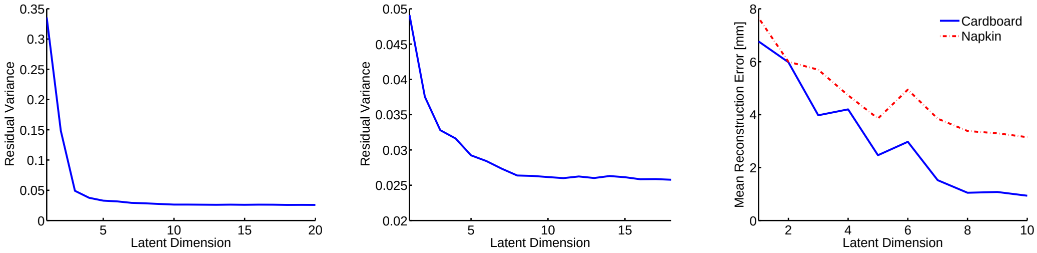


Figure 7.8: Comparison of the different local models. We plot the average reconstruction error on cardboard (left) and napkin (right) test patches. From top to bottom, the models used were obtained from cardboard data, napkin data, synthetic data, and with modal analysis.

require real training data, we could not always apply all of them to all cases. The results nonetheless highlight their strengths and weaknesses.



(a) (b) (c)

Figure 7.9: Learning a non-linear model from synthetic data. (a) Running Isomap on the data gives a residual error that strongly decreases until dimension 4, but that truly stabilizes at dimension 10, as shown in the zoomed-in plot (b). (c) We learned non-linear models for latent dimensions up to 10 and tested them on real cardboard and napkin data. These models achieve poor performance compared to those trained on real data.

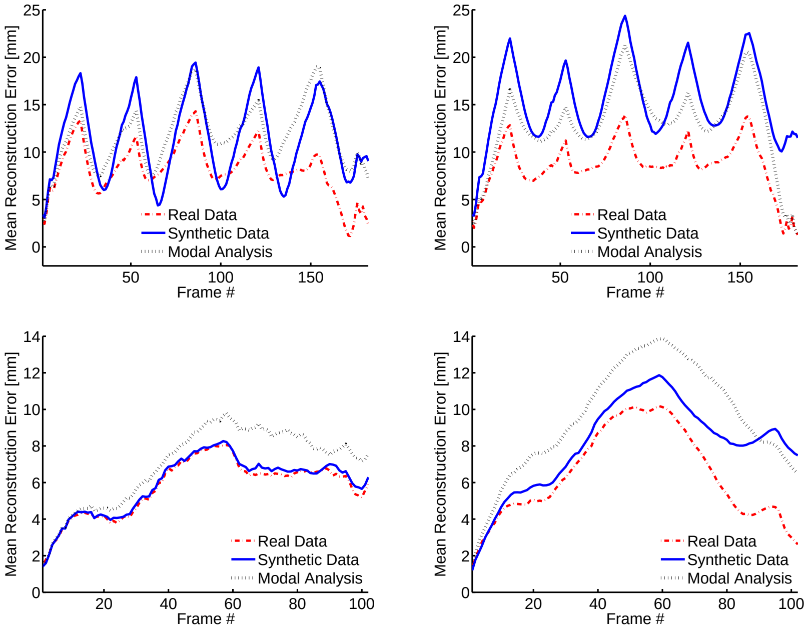


Figure 7.10: Reconstruction errors on synthetic images generated from validation data. We applied a nicely textured image (left) or a much more uniform one (right) to ground-truth meshes, and projected them with known camera. The top row corresponds to the piece of cardboard, and the bottom one to the paper napkin. In each plot, we show the results for the local models obtained from real data (red), synthetic data (blue), and modal analysis (black).

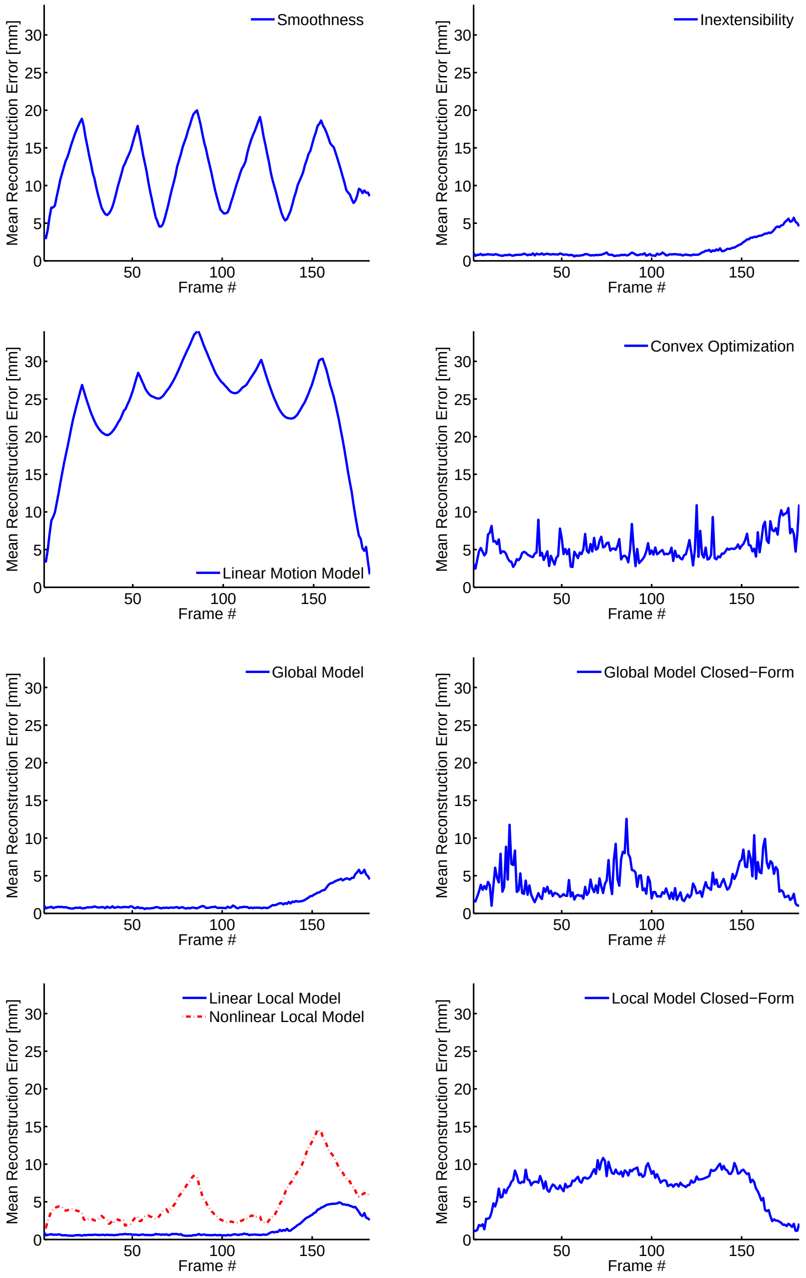


Figure 7.11: Reconstruction errors on synthetic well-textured cardboard images using all the methods discussed in this thesis.

7.3.1 Synthetic Data

We first tested our approaches on the same synthetic images as in the previous section. However, here we applied our techniques as presented in the previous chapters. There-

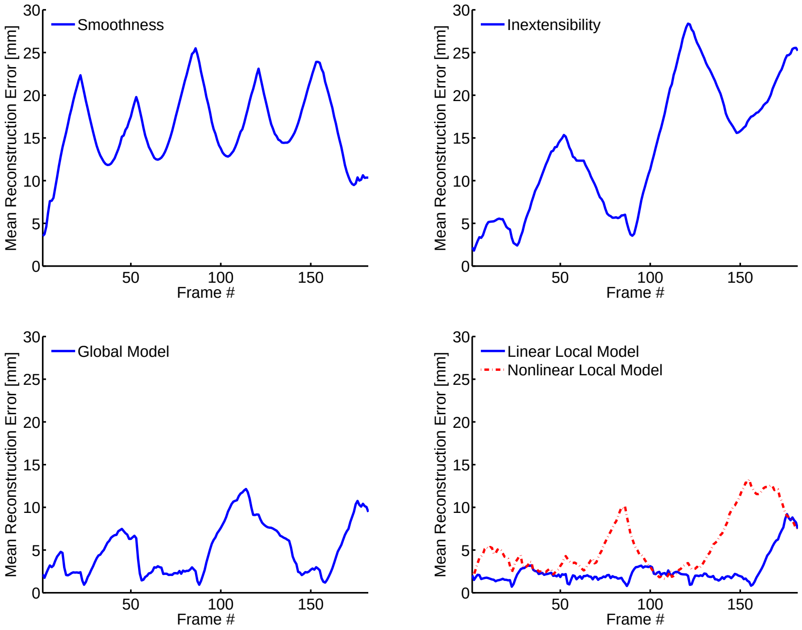


Figure 7.12: Same plots as those of Fig. 7.11, but for much less textured cardboard images. Since some methods rely on correspondences, we could not apply them to this case.

fore, linear models, both global and local, were used in conjunction with inextensibility constraints. Additionally, since preventing the surface to expand or shrink proved useful in many cases, we computed results using inextensibility constraints only. In Figs. 7.11 and 7.12, we show these errors as a function of time for the textured and more uniform cardboard sequences. Figs. 7.13 and 7.14 show similar plots for the paper napkin sequences. Note that methods that rely on correspondences were only evaluated in the textured cases.

These error plots show us that, with a well-textured surface, most methods perform well, with the exception of the linear motion model of Section 4.1.2. This was to be expected, since it really represents the minimal set of constraints to help recovery, and does not truly account for the behavior of deformable surfaces. Similarly, smoothness alone performs poorly, especially on the cardboard case. By contrast, note that inextensibility constraints alone yield a good reconstruction. This implies that the global and local linear models also perform very well.

When dealing with much less-textured surfaces, the results are different. First, the performances of smoothness constraints have further degraded. Second, inextensibility constraints cannot be used on their own anymore, since there is no texture to prevent sharp creases from appearing on the surface. However, when used in conjunction with a deformation model, global or local, they yield good reconstructions.

Furthermore, these experiments show that convex optimization and closed-form methods

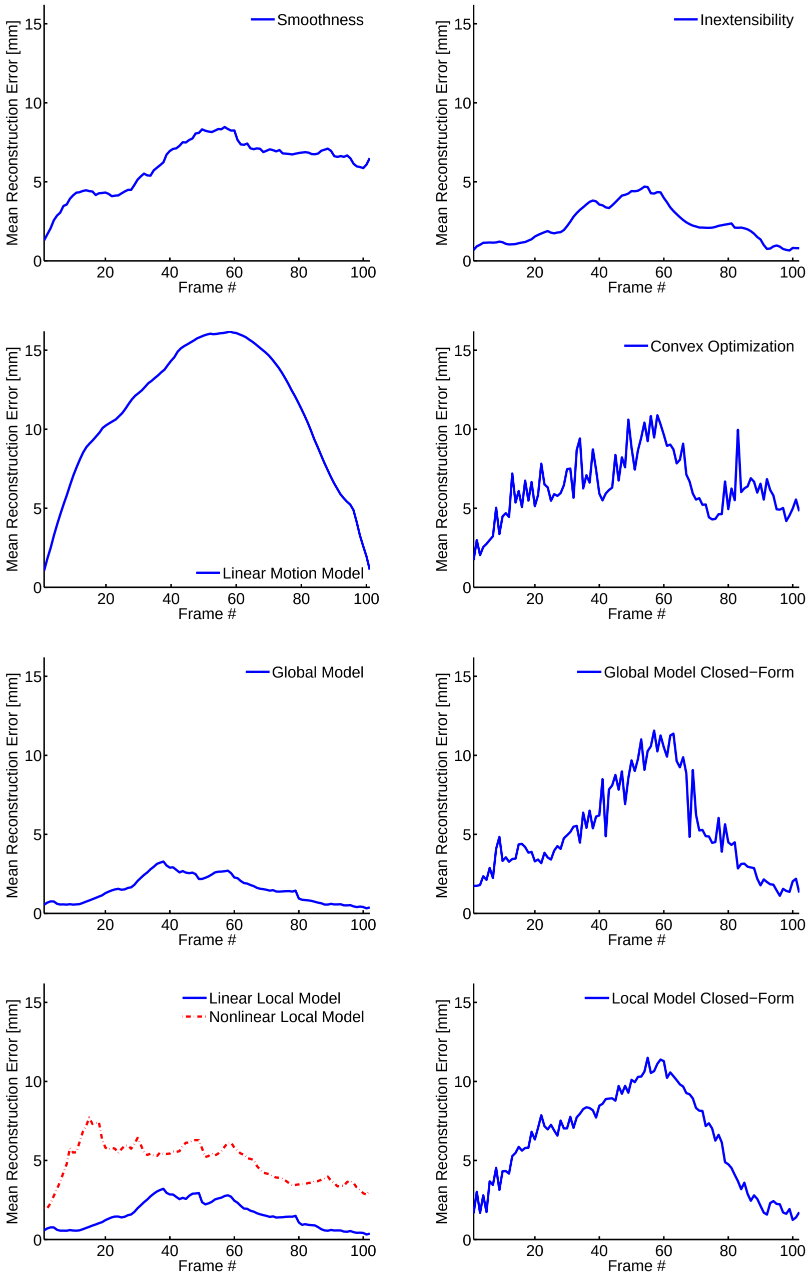


Figure 7.13: Reconstruction errors on synthetic well-textured napkin images. We compare our methods together and with smoothness and inextensibility constraints.

yield very similar results. This seems natural since they all rely on correspondences, and approximate inextensibility constraints, though in different ways: In the SOCP method,

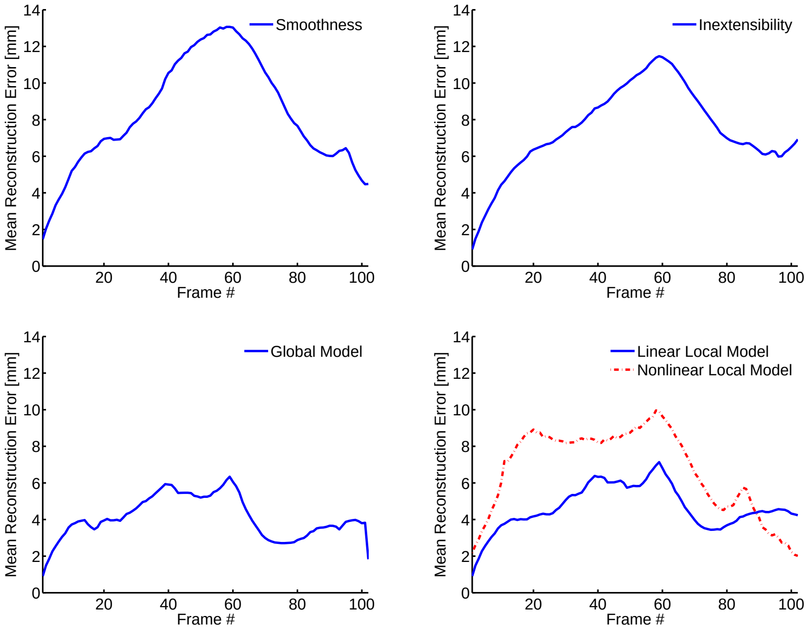


Figure 7.14: Same plots as in Fig. 7.13, but for much less textured napkin images. Since some methods rely on correspondences, we could not apply them to this case.

we ensure that edges lengths will not change by more than 10%, and in the closed-form solutions, we solve linearized quadratic equations in terms of parameters that can only approximately enforce these constraints. As can also be seen from the plots, linear global and local models used in tracking perform better than in closed-form. The fact that they rely on template matching and that an objective function is truly minimized is expected to improve the registration with the image and thus the recovered shape.

An interesting observation is that linear global and local models perform similarly, which might seem surprising. This suggests that our global models have sufficient flexibility to undergo complex deformations, which might seem to make local models unnecessary. Nonetheless, global models can only handle the specific surface for which they have been trained, whereas local ones can be assembled to handle any shape. As will be shown below, we believe this to be their real strength.

Finally, we note that the linear models perform slightly better than the non-linear ones when inextensibility constraints are enforced, but worse when they are not. This makes the non-linear models more attractive when dealing with flexible materials where folds can appear in-between vertices, as has already been shown in Section 6.4.3. Recall, however, that they can only be used for tracking purposes, whereas the linear ones can recontruct surfaces from individual images.

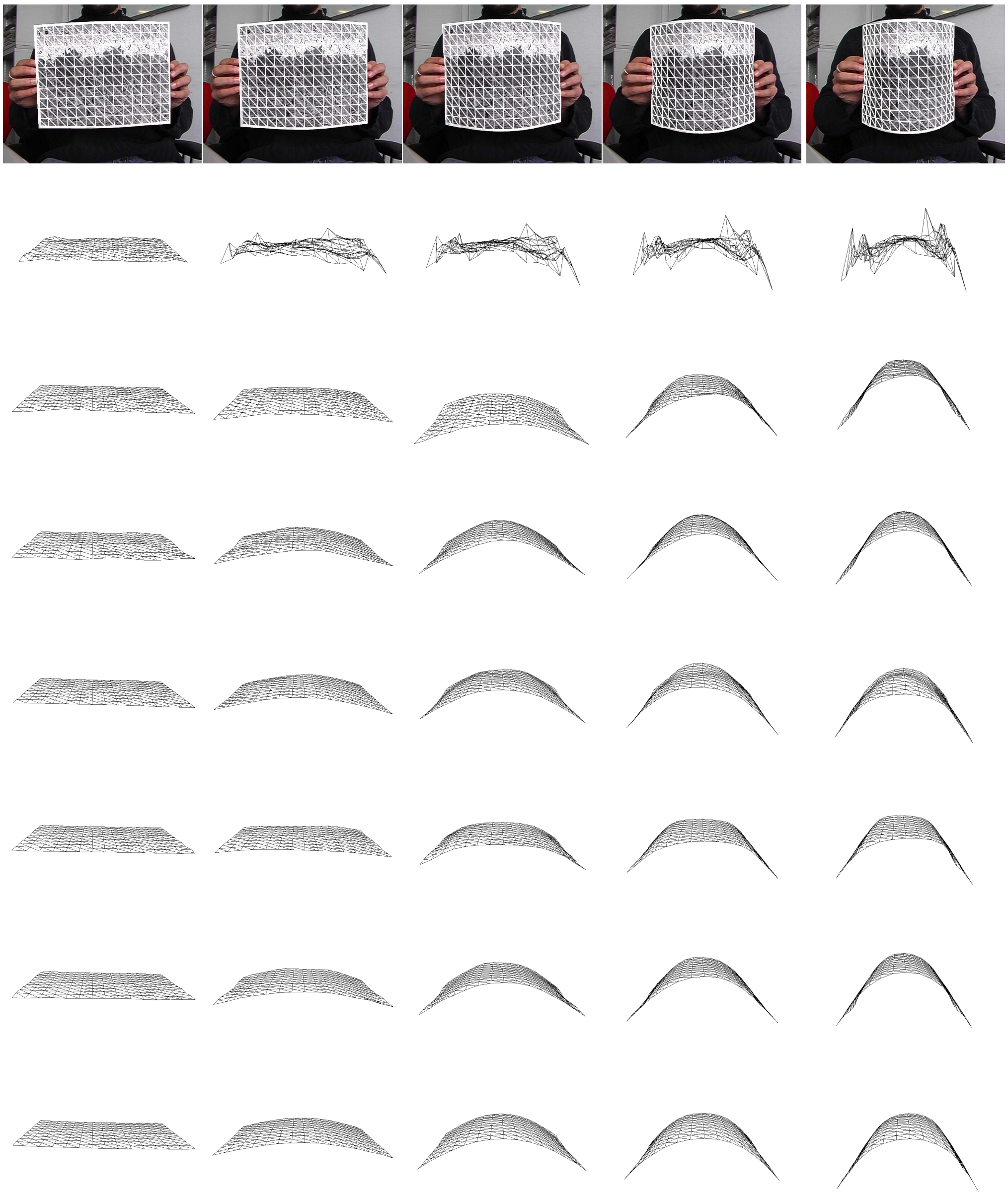


Figure 7.15: We compared our methods on a real video of a deforming sheet of cardboard. In order to texture it, and thus make it usable for correspondences-based techniques, we stuck a piece of paper on the cardboard. Since cardboard is more rigid, its behavior remains unchanged. Top row: Reprojection of the results obtained with the linear local model. Next rows: Results obtained, from top to bottom, with linear motion model, convex optimization, linear global model by tracking and in closed-form, non-linear local models, linear local models by tracking and in closed form.

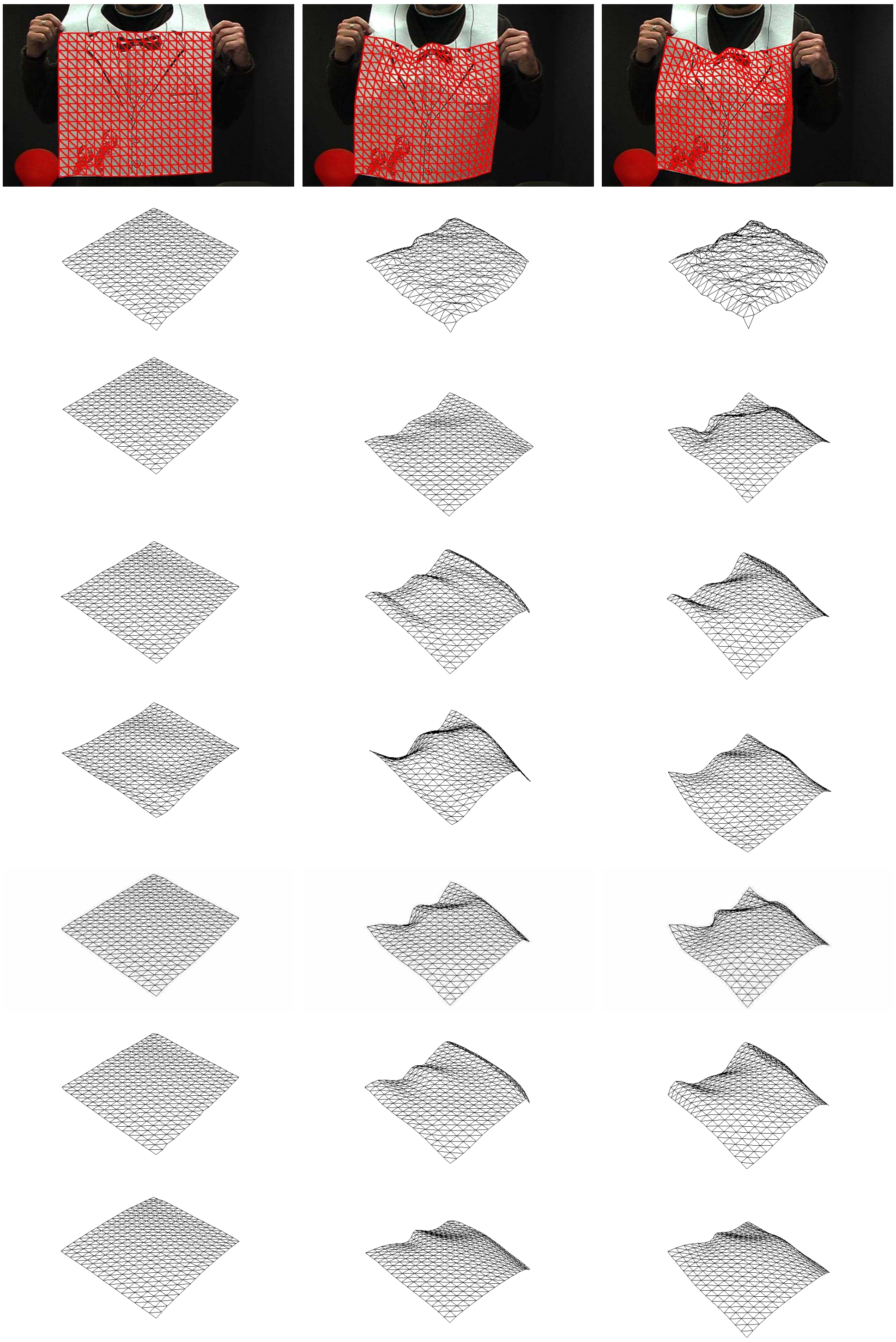


Figure 7.16: Same comparison as in the cardboard case but for the more flexible napkin. In this case, we computed correspondences from the results obtained with the non-linear local models.

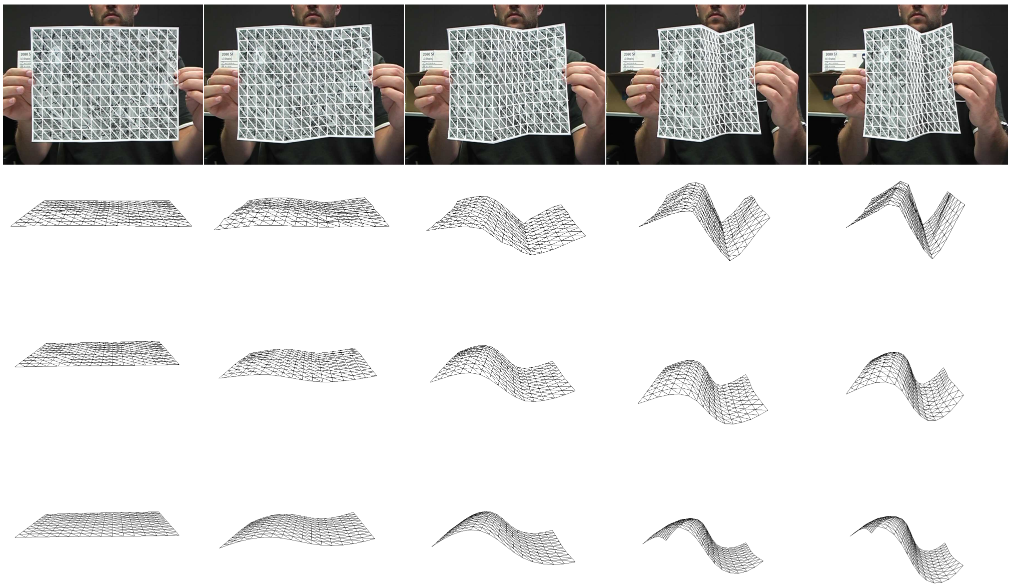


Figure 7.17: Comparing our approaches on a piece of paper with sharp creases. From top to bottom, we show the results of the convex optimization approach and the closed-form solutions of global and local models. Methods relying on a deformation model are not truly adapted to this case, and tend to oversmooth the surface. Nonetheless their results are reasonable approximations of the true shapes.

7.3.2 Real Images

We applied our methods to several real sequences of more or less textured surfaces undergoing large deformations. The surfaces were made of cardboard, paper, and more flexible tissue, which all have very different physical properties. Since some of our approaches rely on correspondences, we needed to have textured surfaces. For the cardboard, we stuck a textured piece of paper on top of the surface. Since cardboard is more rigid, this did not influence its deformations. This allowed us to use our local cardboard model, as well as methods relying on feature points. Results are depicted in Fig. 7.15, whose first row shows the mesh obtained with linear local models reprojected on the original images. The following rows show the results of the different methods seen from a different viewpoint.

As we can see, most models, to the exception of the linear motion model, perform relatively well on this case. The poor results of the linear motion model can be explained by the fact that, in such a deformation, the motion along the line of sight is large. Therefore, a method that penalizes it is not adapted. Note that the non-linear models tend to flatten the top of the bent paper. We believe this to be due to the absence of true inextensibility constraints, which typically would not allow this to happen.

We then wanted to compare our methods on the paper napkin case, which is much more

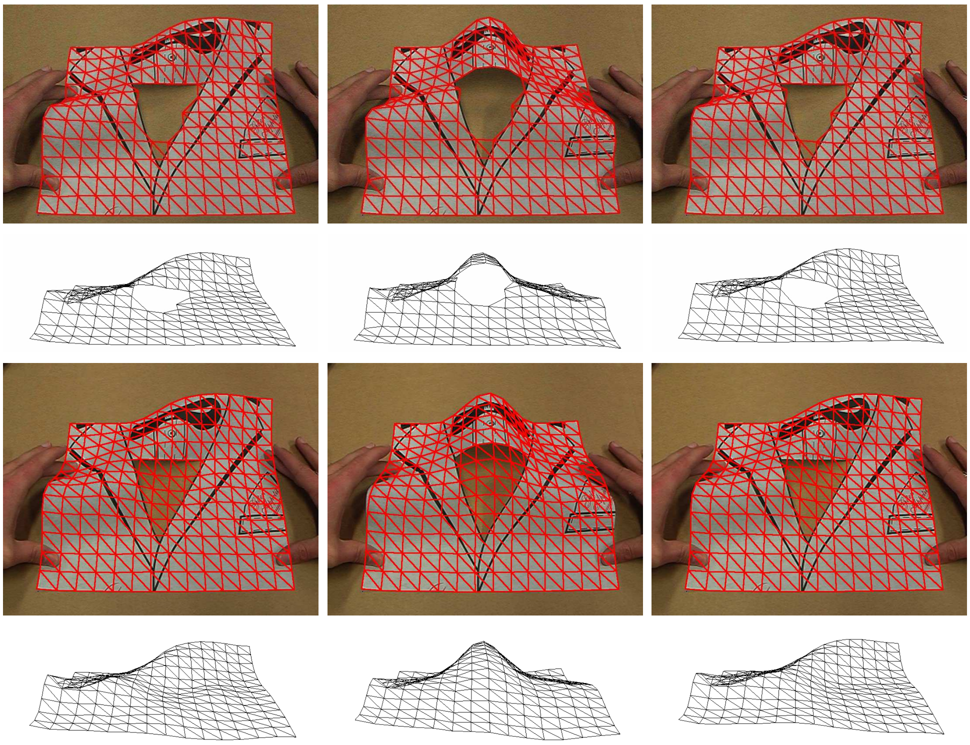


Figure 7.18: We compared the reconstructions with local (top) and global (bottom) models of a surface with a hole. Even though the global model manages to track the surface, it oversmoothes the parts where the hole creates discontinuities. This yields a bad reprojection of the resulting mesh.

flexible. Unfortunately, in this case, we cannot stick texture on it without changing its properties. Therefore, for methods that rely on correspondences, we established them from the results of the non-linear local models, by sampling the barycentric coordinates of the facets and adding gaussian noise with variance 2 to the reprojections of these points. This is a convoluted way of computing correspondences, but, since we only aim at checking whether a model can recover complex deformations, it is reasonable. Similar results as in the cardboard case are depicted in Fig. 7.16. For this experiment, we notice the superiority of the non-linear local models over the other techniques. Similarly, we observe that the convex optimization approach performs well. Linear local and global models have troubles modeling such a flexible surface where inextensibility constraints can be violated since folds may appear between neighboring vertices. The linear motion model again performs worst, though not as dramatically as for the cardboard case.

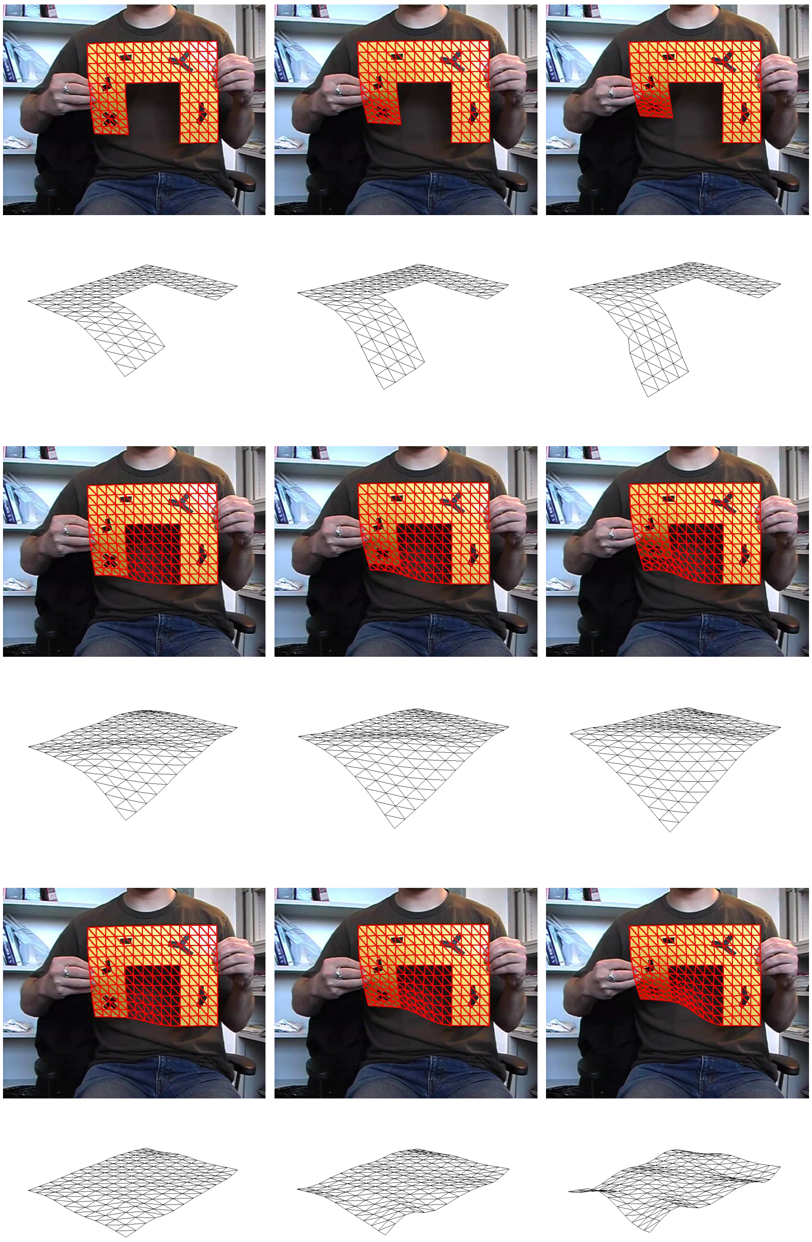


Figure 7.19: Comparison of the local and global models on a piece of cardboard from which a large part was cut out. This simulates the extreme case of a rectangular hole where the two opposite parts can move independently. In the top two rows, we show the results of the local models that correctly recover the shape. In the middle, we display results obtained with a global model for which we penalized stretching. For small deformations, this representation still works, but the two parts cannot be moved far apart, since this would violate inextensibility constraints. Without these constraints, the mesh reprojects correctly on the images, but gives a meaningless 3D shape, as shown in the bottom two rows.

We also evaluated our methods on a sequence of a piece of paper with sharp creases. This corresponds to the ultimate flexible case, and is a challenge for methods that favor smooth results. For this case, we only evaluated the closed-form versions of the linear models, and compared them to the convex optimization technique. As shown in Fig. 7.17, only our convex optimization technique manages to recover such a shape precisely. Nonetheless, both global and local models yield meaningful approximations of the true shape.

Finally, since from results on synthetic data it appeared that optimized global and local models performed equally well, we applied them on surfaces with holes and cuts to demonstrate the advantage of local models over global ones. As can be seen in Fig. 7.18, using a global model to reconstruct the same napkin with a hole as in Section 6.4.3 yields the same kind of error as when we used local models without explicitly accounting for the hole. The global model smoothes the discontinuities created by the hole, which results in a surface that does not reproject correctly. This, nonetheless, only represents a relatively small hole compared to the global surface. To study the case of a much larger hole, we cut out a large piece of a cardboard surface, as shown in Fig. 7.19. This simulates the extreme case where to opposite sides of the hole can move independently. As can be observed from the figure, when penalizing stretching of the mesh, the global model only manages to explain small deformations, and fails to reproject correctly for larger ones, since they would violate inextensibility constraints. One could then think of removing such constraints. In that case, as shown in the last two rows of the figure, the reprojection of the mesh is correct, but its 3D shape completely meaningless.

8 Conclusion

In this thesis, we have presented several solutions to the problem of recovering the 3D shape of a deforming surface from a single viewpoint. We have shown that, when using feature points and representing the surface as a mesh, recovering its shape amounts to solving an ill-conditioned linear system. This is because the distances between the vertices and the camera are only poorly constrained by correspondences, which makes 3D shape recovery much more complex than 2D registration.

This observation lead us to study different approaches to formulating the deformation models required to resolve the ambiguities. In a first approach, we constrained shape recovery by using temporal information. The key observation was that vertices cannot move arbitrarily between the consecutive frames of a video sequence. The resulting two techniques, that is imposing a linear motion model or using Second Order Cone Programming, are extremely easy to apply, but are subject to drift as they rely on frame-to-frame tracking.

In a second approach, we introduced the use of shape models. We first proposed a linear global model and presented a technique to create training shapes automatically by randomly setting values to a set of determining angles between the facets of a mesh. Since this global model was restricted to represent a particular object shape, we introduced local models that can be combined to form any shape, and therefore simulate a particular material rather than a particular surface. In this framework, we studied both linear and non-linear models, the latter providing a more accurate representation of the deformation space. However, when inextensibility constraints can be enforced, both kinds of models perform similarly.

Comparing our models allowed us to draw several conclusions. First, the convex optimization method, assuming enough correspondences are provided, yields very good results. This is especially true for surfaces with sharp folds and creases, which the other approaches tend to oversmooth. Unfortunately, it cannot deal with individual images. The linear models, global or local, let us solve the reconstruction in closed-form. They can therefore work on a single image and require no a priori shape estimate. However, such a closed-form solution still requires texture. Nevertheless, when there are too few keypoints on the surface, these models can still be used for tracking. In terms of performance, the global and local approaches seem to give equivalent results. Nonetheless, the local models let us represent surfaces of arbitrary shapes. The major disadvantage of linear models is that their approximation of the shape space is not as accurate as the non-linear ones. This forced us to use them in conjunction with inextensibility constraints, which may not always be truly valid, as was the case with the napkin. For such cases, the non-linear local models are therefore the most adapted. However, they require real training data and yield a much

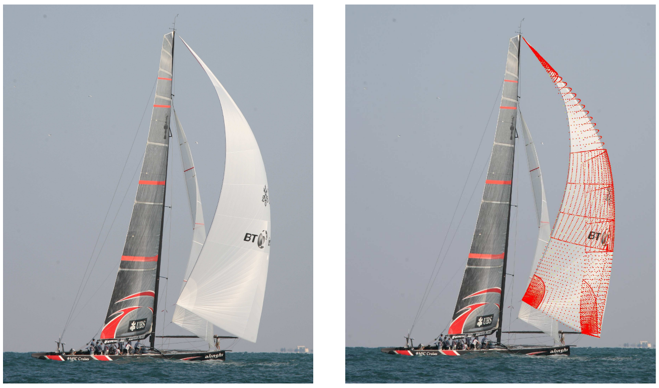


Figure 8.1: Some of our techniques have been used in a software that computes the shape of the spinnaker of the swiss sailing team Alinghi.

more complex objective function, which only makes them practical for tracking purposes.

In short, local models should generally be favored, as long as no sharp creases appear on the surface. Given a video sequence and real training data, the non-linear models are more appropriate. However, when these two conditions are not satisfied, the linear ones are most adapted.

8.1 Applications

As mentioned in the introduction, 3D deformable surface reconstruction is applicable in domains such as medical imaging, entertainment, or sports. Some of our techniques have already been used for the latter. In a collaboration with the swiss sailing team Alinghi, we have developed the Deform3D software that integrates Julien Pilet's image correspondence technique [128] and reconstructs the shape of a spinnaker from a series of images, such as the ones of Fig. 8.1. As depicted by Fig. 8.2, our methods have also been used to recover the shape of the main sail of the french boat Hydroptère [71], shown in Fig. 8.3.

Several other applications following similar ideas are currently under developement. Among them, we study the problem of recovering the shape of the wings of a plane in flight. We also consider applying our techniques to recover the deformations of the rotating turbine blades of a plane. Finally, in medical imaging, our methods could be of interest for laparoscopic surgery. First, they could be used to reconstruct the shape of organs from the noisy laparoscope images. Second, from an external camera, they could help recovering the motion of the patient's torso, from which the deformations of the organs can be infered. We are currently trying to find partners in the medical world to put these ideas into practice.

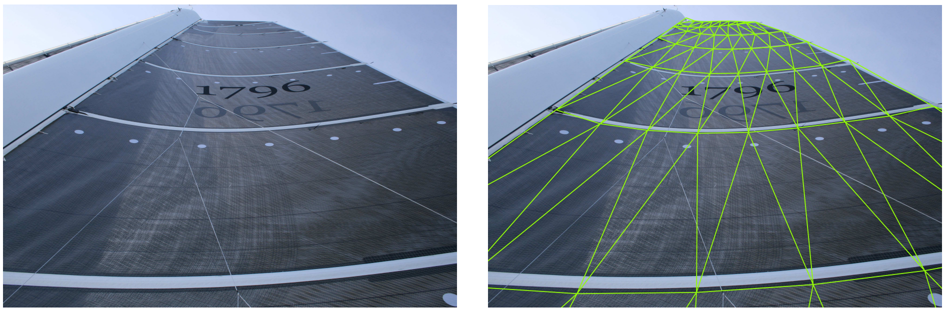


Figure 8.2: Similarly as for the spinnaker, our methods were applied to recover the shape of the main sail of the french boat Hydroptère.

8.2 Future Work

Even though the methods developped in this thesis have advanced the state-of-the-art in deformable 3D shape recovery, many further improvements are possible. One of the major limitations of our approaches is their need of a reference image in which we know the shape of the surface. In many real life applications, such an image is not available and we only have images of the deformed surface. This problem is related to non-rigid structure-from-motion. However, the current solutions require many frames and only recover relatively small deformations. Furthermore, they only reconstruct a few 3D points rather than the whole surface. A challenging topic for future research would be to study the ambiguities when given two images of the same surface undergoing two different deformations. One could reasonably further assume that a 3D model is known, but does not correspond to either images. Given these ambiguities, one could study the appropriate constraints to solve them, and possibly use the ones developped in this work. The main challenge in solving this problem stems from the fact that we do not know to which facet of the mesh a feature point detected on the image belongs. This would create integer programming problems with new unknowns for the facets indices.

In a more straightforward direction, one could study more deeply the inextensibility constraints. As shown in this thesis, they are very good at selecting plausible solutions of an ill-conditioned linear system. However, by themselves, they cannot resolve all the ambiguities. This was observed in the case of human motion, where one can flip in depth the links between the joints of a skeleton and still obtain a correct reprojection [142]. In that work, the ambiguities were solved by exploring all possible flips of the links, and choosing the most appropriate ones. In our case, the problem becomes far more complex, since we are dealing with complete surfaces as opposed to joints that form a simple tree structure. Nonetheless, we could find the multiple minima of the quadratic equations accounting for edges lengths that appear in our closed-form solutions rather than one solution, as is the case with Extended Linearization. For small meshes, this could be achieved using Groebner bases. For larger surfaces, the problem has too many unknowns for Groebner bases to be practical, however, one could consider finding the multiple solutions of small parts of the

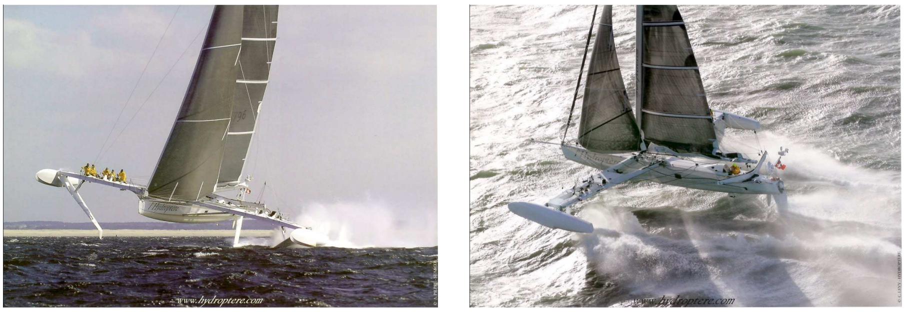


Figure 8.3: Images of the french boat Hydroptère [71].

mesh, and link them together in a consistent manner.

This would give us a way to exploit shading, an important image cue that we have neglected in this work. As mentioned in Chapter 2, it gives a good idea of the shape of a surface, but relies on very strong assumptions that almost never hold. To make it practical, it would seem more appopriate to use it as a qualitative measure rather than a quantitative one. Given the multiple results obtained with inextensibility contraints as described above, one could then use shading to select the correct one. In such conditions, an approximate shading model should be sufficient, and should let us find the correct shape even when the assumptions made by the model are violated. The selection could be done by choosing the shape that yields maximum mutual information with the image, as was proposed in [166].

Finally, another avenue of research arises from the formulation of our problem in a different metric space than the standard Euclidean space. Recently, a non-rigid structure-from-motion method formulated as an optimization problem in a Riemannian manifold has been proposed [139]. The main advantage of using such a manifold is that it allows to define geodesic distances and isometric deformations. This was first introduced in Computer Graphics for simulation purposes [86]. Following this approach, we could implicitely enforce inextensibility constraints without the need of non-convex constraints. However, the drawback of the current formulations is that they relate deformations over time, and thus are inappropriate for shape recovery from a single image.

Bibliography

[1] AANAES, H., AND KAHL, F. Estimation of deformable structure and motion. In *Vision and Modelling of Dynamic Scenes Workshop* (2002).

[2] AGARWAL, A., AND TRIGGS, B. 3d human pose from silhouettes by relevance vector regression. In *Conference on Computer Vision and Pattern Recognition* (2004).

[3] AHMED, A., AND FARAG, A. A new formulation for shape from shading for non-lambertian surfaces. In *Conference on Computer Vision and Pattern Recognition* (New York, June 2006).

[4] ANDRILUKA, M., ROTH, S., AND SCHIELE, B. People-tracking-by-detection and people-detection-by-tracking. In *Conference on Computer Vision and Pattern Recognition* (Anchorage, Alaska, June 2008).

[5] ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. Scape: shape completion and animation of people. *ACM SIGGRAPH 24* (2005), 408–416.

[6] BALAN, A. O., BLACK, M. J., HAUSSECKER, H. W., AND SIGAL, L. Shining a light on human pose: On shadows, shading and the estimation of pose and shape. In *International Conference on Computer Vision* (October 2007).

[7] BALAN, A. O., SIGAL, L., BLACK, M. J., DAVIS, J. E., AND HAUSSECKER, H. W. Detailed human shape and pose from images. In *Conference on Computer Vision and Pattern Recognition* (June 2007).

[8] BARAFF, D., AND WITKIN, A. Large steps in cloth simulation. In *ACM SIGGRAPH* (1998), pp. 43–54.

[9] BARBIC, J., AND JAMES, D. Real-time subspace integration for st.venant-kirchhof deformable models. In *ACM SIGGRAPH* (2005).

[10] BARTOLI, A., AND OLSEN, S. A Batch Algorithm For Implicit Non-Rigid Shape and Motion Recovery. In *ICCV Workshop on Dynamical Vision* (Beijing, China, October 2005).

[11] BARTOLI, A., AND ZISSERMAN, A. Direct Estimation of Non-Rigid Registration. In *British Machine Vision Conference* (Kingston, UK, September 2004).

[12] BATHE, K.-J. *Finite Element Procedures in Engineering Analysis*. Prentice Hall, 1982.

[13] BELHUMEUR, P., KRIEGMAN, D., AND YUILLE, A. The Bas-Relief Ambiguity. *International Journal of Computer Vision 35*, 1 (1999), 33–44.

[14] BELKIN, M., AND NIYOGI, P. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2001, pp. 585–591.

[15] BHAT, K. S., TWIGG, C. D., HODGINS, J. K., KHOSLA, P. K., POPOVIC, Z., AND SEITZ, S. M. Estimating cloth simulation parameters from video. In *ACM Symposium on Computer Animation* (2003).

[16] BLACK, M. J., AND JEPSON, A. D. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conference on Computer Vision* (1996), pp. 329–342.

[17] BLAKE, A., ZIMMERMAN, A., AND KNOWLES, G. Surface descriptions from stereo and shading. *Image Vision Computing 3*, 4 (1986), 183–191.

[18] BLANZ, V., BASSO, C., POGGIO, T., AND VETTER, T. Reanimating Faces in Images and Video. In *Eurographics* (Granada, Spain, September 2003).

[19] BLANZ, V., AND VETTER, T. A Morphable Model for The Synthesis of 3–D Faces. In *ACM SIGGRAPH* (Los Angeles, CA, August 1999), pp. 187–194.

[20] BOTSCH, M., PAULY, M., WICKE, M., AND GROSS, M. Adaptive space deformations based on rigid cells. In *Eurographics* (2007).

[21] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, 2004.

[22] BOYER, E., AND BERGER, M.-O. 3D Surface Reconstruction Using Occluding Contours. *International Journal of Computer Vision 22*, 3 (1997), 219–233.

[23] BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. Markerless garment capture. *ACM Trans. Graph. 27*, 3 (2008).

[24] BRAND, M. Morphable 3d models from video. *Conference on Computer Vision and Pattern Recognition* (2001).

[25] BREGLER, C., HERTZMANN, A., AND BIERMANN, H. Recovering non-rigid 3d shape from image streams. In *Conference on Computer Vision and Pattern Recognition* (2000).

[26] BRIDSON, R., FEDKIW, R., AND ANDERSON, J. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (ToG)* (2002), pp. 594–603.

[27] BRIDSON, R., MARINO, S., AND FEDKIW, R. Simulation of clothing with folds and wrinkles. In *ACM Symposium on Computer Animation* (2003).

[28] CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 6 (1986).

[29] CARCERONI, R. L., AND KUTULAKOS, K. N. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *International Journal of Computer Vision 49*, 2-3 (2002), 175–214.

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

[31] CHANDRAKER, M., KAHL, F., AND KRIEGMAN, D. Reflections on the generalized bas-relief ambiguity. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005).

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

[33] CIRAK, F., ORTIZ, M., AND SCHRÖDER, P. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering 47* (2000), 2039–2072.

[34] 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.

[35] COHEN, L., AND COHEN, I. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 11 (November 1993), 1131–1147.

[36] COMON, P. Independent component analysis, a new concept? *Signal Processing 36*, 3 (1994), 287–314.

[37] COOTES, T., EDWARDS, G., AND TAYLOR, C. Active Appearance Models. In *European Conference on Computer Vision* (Freiburg, Germany, June 1998), pp. 484–498.

[38] COOTES, T. F., AND TAYLOR, C. J. Active shape models - 'smart snakes. In *British Machine Vision Conference* (1992), pp. 266–275.

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

[40] COURTOIS, N., KLIMOV, A., PATARIN, J., AND SHAMIR, A. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT* (Bruges, Belgium, May 2000).

[41] CROSS, G., AND ZISSERMAN, A. Surface reconstruction from multiple views using apparent contours and surface texture. In *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics, Ljubljana, Slovenia* (2000), A. Leonardis, F. Solina, and R. Bajcsy, Eds., pp. 25–47.

[42] DE AGUIAR, E., THEOBALT, C., STOLL, C., AND SEIDEL, H.-P. Marker-less deformable mesh tracking for human shape and motion capture. In *Conference on Computer Vision and Pattern Recognition* (Minneapolis, USA, June 2007).

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

[44] 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).

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

[46] DRBOHLAV, O., AND SÁRA, R. Specularities reduce ambiguity of uncalibrated photometric stereo. In *European Conference on Computer Vision* (2002), pp. 46–62.

[47] DYN, N., LEVINE, D., AND GREGORY, J. A. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics 9*, 2 (April 1990), 160–169.

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

[49] FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Transactions on Visualization and Computer Graphics* (1997).

[50] FORSYTH, D., AND ZISSERMAN, A. Reflections on shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 7 (July 1991), 671–679.

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

[52] FUA, P., AND LECLERC, Y. G. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision 16* (September 1995), 35–56.

[53] GAY-BELLILE, V., BARTOLI, A., AND SAYD, P. Direct estimation of non-rigid registrations with image-base self-occlusion reasoning. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).

[54] GREMINGER, M. A., AND NELSON, B. J. Deformable object tracking using the boundary element method. In *Conference on Computer Vision and Pattern Recognition* (2003).

[55] GREMINGER, M. A., AND NELSON, B. J. A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges. *International Journal of Computer Vision 78*, 1 (2008), 29–45.

[56] GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. Discrete shells. In *ACM Symposium on Computer Animation* (2003), pp. 62–67.

[57] GROCHOW, K., MARTIN, S., HERTZMANN, A., AND POPOVIC, Z. Style-based inverse kinematics. In *ACM SIGGRAPH* (2004), pp. 522–531.

[58] GUMEROV, N., ZANDIFAR, A., DURAISWAMI, R., AND DAVIS, L. Structure of Applicable Surfaces from Single Views. In *European Conference on Computer Vision* (Prague, May 2004).

[59] HARRIS, C., AND STEPHENS, M. A Combined Corner and Edge Detector. In *Fourth Alvey Vision Conference, Manchester* (1988).

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

[61] HAUTH, M., AND STRASSER, W. Corotational simulation of deformable solids. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2004), pp. 137–145.

[62] HERNANDEZ, C., VOGIATZIS, G., BROSTOW, G. J., STENGER, B., AND CIPOLLA, R. Non-rigid photometric stereo with colored lights. In *International Conference on Computer Vision* (October 2007).

[63] HERTZMANN, A., AND SEITZ, S. M. Shape and materials by example: A photometric stereo approach. In *Conference on Computer Vision and Pattern Recognition* (2003), pp. 533–540.

[64] HINTON, G. E. Products of experts. In *International Conference on Artificial Neural Networks (ICANN)* (1999), pp. 1–6.

[65] HIROTA, G., FISHER, S., AND LIN, M. Simulation of non-penetrating elastic bodies using distance fields. Tech. rep., University of North Carolina, Chapel Hill, NC, USA, 2000.

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

[67] HORN, B., AND BROOKS, M. *Shape from Shading*. MIT Press, 1989.

[68] HOTELLING, H. Relations between two sets of variables. *Biometrika 28*, 3 (December 1936), 321–377.

[69] HOUSE, D. H., AND BREEN, D. E., Eds. *Cloth modeling and animation*. A. K. Peters, Ltd., 2000.

[70] HUANG, W., GOLDGOF, D., AND TSAP, L. Nonlinear finite element methods for nonrigid motion analysis. In *Physics-Based Modelling in Computer Vision* (1995).

[71] Hydroptere. http://www.hydroptere.com/.

[72] 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).

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

[74] ILIĆ, S., AND FUA, P. Implicit Meshes for Surface Reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 8*, 2 (2006), 328–333.

[75] ILIC, S., AND FUA, P. Non-Linear Beam Model for Tracking Large Deformation. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).

[76] 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).

[77] ILIĆ, S., SALZMANN, M., AND FUA, P. Implicit Meshes for Effective Silhouette Handling. *International Journal of Computer Vision 72*, 7 (2007).

[78] IRVING, G., TERAN, J., AND FEDKIW, R. Invertible finite elements for robust simulation of large deformation. In *Symposium on Computer Animation* (2004).

[79] JAMES, D. L., AND PAI, D. K. Artdefo: accurate real time deformable objects. In *ACM SIGGRAPH* (1999), pp. 65–72.

[80] JOJIC, N., AND HUANG, T. S. Estimating cloth draping parameters from range data. In *Workshop on Synthetic-Natural Hybrid Coding and 3-D Imaging* (1997), pp. 73–76.

[81] JOLLIFFE, I. T. *Principal Component Analysis*. Springer-Verlag, 1986.

[82] KAHL, F. Multiple view geometry and the L_∞-norm. In *International Conference on Computer Vision* (Beijing, China, 2005), pp. 1002–1009.

[83] KAMBHAMETTU, A., GOLDGOFF, D., TERZOPOULOS, D., AND HUANG, T. *Handbook of Pattern Recognition and Image Processing: Computer Vision*. Academic Press, 1994, ch. Non Rigid Motion Analysis, pp. 405–430.

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

[85] KE, Q., AND KANADE, T. Quasiconvex optimization for robust geometric reconstruction. In *International Conference on Computer Vision* (2005), pp. 986–993.

[86] KILIAN, M., MITRA, N. J., AND POTTMAN, H. Geometric modeling in shape space. In *ACM SIGGRAPH* (2007).

[87] KOBBELT, L. P. Sqrt(3)-subdivision. In *ACM SIGGRAPH* (2000).

[88] KOBBELT, L. P., BAREUTHER, T., AND PETER SEIDEL, H. Multiresolution shape deformations for meshes with dynamic vertex connectivity. In *Eurographics* (2000).

[89] KRIEGMAN, D. J., AND BELHUMEUR, P. N. What shadows reveal about object structure. In *European Conference on Computer Vision* (1998), pp. 399–414.

[90] KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygon meshes. In *ACM SIGGRAPH* (1996), pp. 313–324.

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

[92] KWON, Y. W., AND BANG, H. *The Finite Element Method Using MATLAB*. CRC Press, 2000.

[93] LAWRENCE, C., ZHOU, J. L., AND TITS, A. L. User's guide for cfsqp version 2.5: A c code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints.

[94] LAWRENCE, N. D. Gaussian Process Models for Visualisation of High Dimensional Data. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2004.

[95] LAWRENCE, N. D. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research 6* (2005), 1783–1816.

[96] LAWRENCE, N. D. Learning for Larger Datasets with the Gaussian Process Latent Variable Model. In *International Workshop on Artificial Intelligence and Statistics* (San Juan, Puerto Rico, 2007).

[97] LAWRENCE, N. D., AND MOORE, A. J. Hierarchical gaussian process latent variable models. In *International Conference in Machine Learning* (2007), pp. 481–488.

[98] LECLERC, Y. G., AND BOBICK, A. F. The Direct Computation of Height from Shading. In *Conference on Computer Vision and Pattern Recognition* (Lahaina, Maui, Hawaii, June 1991).

[99] LEPETIT, V., AND FUA, P. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 9 (Sept. 2006), 1465–1479.

[100] LIANG, J., DEMENTHON, D., AND DOERMANN, D. Flattening curved documents in images. In *Conference on Computer Vision and Pattern Recognition* (2005), pp. 338–345.

[101] LLADO, X., BUE, A. D., AND AGAPITO, L. Non-rigid 3D Factorization for Projective Reconstruction. In *British Machine Vision Conference* (Oxford, UK, September 2005).

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

[103] LOWE, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 20*, 2 (2004), 91–110.

[104] MATTHEWS, I., AND BAKER, S. Active Appearance Models Revisited. *International Journal of Computer Vision 60* (November 2004), 135–164.

[105] MCINERNEY, T., AND TERZOPOULOS, D. A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking. In *International Conference on Computer Vision* (Berlin, Germany, 1993), pp. 518–523.

[106] MCINERNEY, T., AND TERZOPOULOS, D. A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4d image analysis. *Computerized Medical Imaging and Graphics 19*, 1 (1995), 69–83.

[107] MCINERNEY, T., AND TERZOPOULOS, D. Deformable models in medical image analysis: A survey. *Medical Image Analysis 1* (1996), 91–108.

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

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

[110] MONTAGNAT, J., DELINGETTE, H., AND AYACHE, N. A review of deformable surfaces: Topology, geometry and deformation. *Image and Vision Computing 19* (2001), 1023–1040.

[111] MORENO-NOGUER, F., LEPETIT, V., AND FUA, P. Accurate Non-Iterative $O(n)$ Solution to the PnP Problem. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).

[112] MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. Stable real-time deformations. In *ACM Symposium on Computer Animation* (2002), pp. 49–54.

[113] NASTAR, C. Vibration modes for nonrigid motion analysis in 3d images. In *European Conference on Computer Vision* (1994).

[114] NASTAR, C., AND AYACHE, N. Fast segmentation, tracking, and analysis of deformable objects. In *International Conference on Computer Vision* (1993), pp. 275–279.

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

[116] NASTAR, C., MOGHADDAM, B., AND PENTLAND, A. Generalized image matching: Statistical learning of physically-based deformations. In *European Conference on Computer Vision* (1996), pp. 589–598.

[117] NAVARATNAM, R., FITZGIBBON, A., AND CIPOLLA, R. The Joint Manifold Model for Semi-supervised Multi-valued Regression. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).

[118] NAYAR, S., IKEUCHI, K., AND KANADE, T. Shape from interreflections. *International Journal of Computer Vision 6*, 3 (1991), 173–195.

[119] NG, H. N., AND GRIMSDALE, R. L. Computer graphics techniques for modeling cloth. *Computer Graphics and Applications 16*, 5 (1996), 28–41.

[120] OREN, M., AND NAYAR, S. Generalization of the Lambertian Model. In *DARPA Image Understanding Workshop* (April 1993), Morgan Kaufmann, pp. 1037–1048.

[121] OREN, M., AND NAYAR, S. A Theory of Specular Surface Geometry. *International Journal of Computer Vision 24*, 2 (1996), 105–124.

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

[123] PENTLAND, A., AND SCLAROFF, S. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13* (1991), 715–729.

[124] PERRIOLLAT, M., AND BARTOLI, A. A quasi-minimal model for paper-like surfaces. In *BenCos Workshop at CVPR'07* (2007).

[125] PERRIOLLAT, M., HARTLEY, R., AND BARTOLI, A. Monocular template-based reconstruction of inextensible surfaces. In *British Machine Vision Conference* (2008).

[126] PICINBONO, G., DELINGETTE, H., AND AYACHE, N. Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. In *International Conference on Medical Robotics, Imaging And Computer Assisted Surgery* (2000), pp. 643–652.

[127] PILET, J., LEPETIT, V., AND FUA, P. Real-Time Non-Rigid Surface Detection. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005).

[128] PILET, J., LEPETIT, V., AND FUA, P. Fast Non-Rigid Surface Detection, Registration and Realistic Augmentation. *International Journal of Computer Vision 76*, 2 (February 2008).

[129] PRADOS, E., AND FAUGERAS, O. Shape from shading: a well-posed problem ? In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005).

[130] REITINGER, B., BORNIK, A., BEICHEL, R., AND SCHMALSTIEG, D. Liver surgery planning using virtual reality. *IEEE Computer Graphics and Applications 26*, 6 (2006), 36–47.

[131] ROMDHANI, S., AND VETTER, T. Efficient, robust and accurate fitting of a 3d morphable model. In *International Conference on Computer Vision* (Nice, France, 2003).

[132] ROWEIS, S., AND SAUL, L. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[133] SAMARAS, D., AND METAXAS, D. Incorporating Illumination Constraints in Deformable Models. In *Conference on Computer Vision and Pattern Recognition* (Santa Barbara, June 1998), pp. 322–329.

[134] SCHOELKOPF, B., BURGES, C., AND SMOLA, A. Advances in kernel methods. In *Support Vector Learning*. MIT Press, 1999.

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

[136] Seidel, R. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry 1* (1991), 51–64.

[137] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Conference on Computer Vision and Pattern Recognition* (2006), pp. 519–528.

[138] Shahrokni, A., Drummond, T., and Fua, P. Fast texture-based tracking and delineation using texture entropy. In *International Conference on Computer Vision* (Beijing, China, October 2005).

[139] Shaji, A., and Chandran, S. Riemannian manifold optimisation for non-rigid structure from motion. In *Conference on Computer Vision and Pattern Recognition* (Anchorage, Alaska, 2008).

[140] Sidenbladh, H., Black, M. J., and Fleet, D. J. Stochastic Tracking of 3D human Figures using 2D Image Motion. In *European Conference on Computer Vision* (June 2000).

[141] Sim, K., and Hartley, R. Removing outliers using the L_∞ norm. In *Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2006), pp. 485–494.

[142] Sminchisescu, C., and Triggs, B. Kinematic Jump Processes for Monocular 3D Human Tracking. In *Conference on Computer Vision and Pattern Recognition* (Madison, WI, June 2003), vol. I, p. 69.

[143] Snelson, E., and Ghahramani, Z. Sparse gaussian processes using pseudo-inputs. In *Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2006.

[144] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. Laplacian surface editing. In *Symposium on Geometry Processing* (2004), pp. 175–184.

[145] Starck, J., and Hilton, A. Spherical matching for temporal correspondence of non-rigid surfaces. In *International Conference on Computer Vision* (2005), pp. 1387–1394.

[146] Starck, J., and Hilton, A. Correspondence labelling for wide-timeframe free-form surface matching. In *International Conference on Computer Vision* (2007).

[147] Sturm, J. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, 1999.

[148] SZELISKI, R., AND WEISS, R. Robust Shape Recovery from Occluding Contours Using a Linear Smoother. *International Journal of Computer Vision 28*, 1 (1998), 27–44.

[149] TENENBAUM, J., DE SILVA, V., AND LANGFORD, J. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[150] 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.

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

[152] TERZOPOULOS, D., WITKIN, A., AND KASS, M. Constraints on deformable models: recovering 3d shape and nongrid motion. *Artificial Intelligence 36*, 1 (1988), 91–123.

[153] TIPPING, M., AND BISHOP, C. Probabilistic Principal Component Anlaysis. *Journal of the Royal Statistical Society, B 6*, 3 (1999), 611–622.

[154] TORRESANI, L., HERTZMANN, A., AND BREGLER, C. Learning non-rigid 3d shape from 2d motion. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2003.

[155] TORRESANI, L., HERTZMANN, A., AND BREGLER, C. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 5 (2008), 878–892.

[156] TORRESANI, L., YANG, D. B., ALEXANDER, E. J., AND BREGLER, C. Tracking and modeling non-rigid objects with rank constraints. In *Conference on Computer Vision and Pattern Recognition* (2001), pp. 493–500.

[157] TSAP, L., GOLDGOF, D., SARKAR, S., AND HUANG, W. Efficient nonlinear finite element modeling of nonrigid objects via optimization of mesh models. *Computer Vision and Image Understanding 69*, 3 (March 1998), 330–350.

[158] TSAP, L. V., GOLDGOF, D. B., AND SARKAR, S. Nonrigid motion analysis based on dynamic refinement of finite element models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 5 (2000), 526–543.

[159] URTASUN, R., FLEET, D., AND FUA, P. Monocular 3–d tracking of the golf swing. In *Conference on Computer Vision and Pattern Recognition* (San Diego, CA, June 2005).

[160] URTASUN, R., FLEET, D., AND FUA, P. 3D People Tracking with Gaussian Process Dynamical Models. In *Conference on Computer Vision and Pattern Recognition* (New York, 2006).

[161] URTASUN, R., FLEET, D., HERTZMAN, A., AND FUA, P. Priors for people tracking from small training sets. In *International Conference on Computer Vision* (Beijing, China, October 2005).

[162] VAILLANT, R., AND FAUGERAS, O. Using Extremal Boundaries for 3D Object Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Feb. 1992).

[163] VEDULA, S., BAKER, S., RANDER, P., COLLINS, R., AND KANADE, T. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 3 (2005), 475–480.

[164] VIDAL, R., AND HARTLEY, R. Perspective nonrigid shape and motion recovery. In *European Conference on Computer Vision* (Marseille, France, October 2008).

[165] VIJAYAKUMAR, B., KRIEGMAN, D. J., AND PONCE, J. Structure and motion of curved 3d objects from monocular silhouettes. In *Conference on Computer Vision and Pattern Recognition* (1996), pp. 327–334.

[166] VIOLA, P., AND WELLS, W. M. Alignment by maximization of mutual information. *International Journal of Computer Vision 24*, 2 (1997), 134–154.

[167] VOLINO, P., COURCHESNE, M., AND MAGNENAT-THALMANN, N. Versatile and efficient techniques for simulating cloth and other deformable objects. In *ACM SIGGRAPH* (August 1995), pp. 137–144.

[168] VOLINO, P., AND MAGNENAT-THALMANN, N. Comparing efficiency of integration methods for cloth simulation. In *Computer Graphics International* (2001), pp. 265–274.

[169] WANG, J., FLEET, D., AND HERTZMANN, A. Gaussian Process Dynamical Models. In *Neural Information Processing Systems* (2005).

[170] WELCH, W., AND WITKIN, A. Free-form shape design using triangulated surfaces. In *ACM SIGGRAPH* (1994), pp. 247–256.

[171] WHITE, R., CRANE, K., AND FORSYTH, D. A. Capturing and animating occluded cloth. In *ACM SIGGRAPH* (2007).

[172] WHITE, R., AND FORSYTH, D. Combining cues: Shape from shading and texture. In *Conference on Computer Vision and Pattern Recognition* (2006).

[173] WHITE, R., AND FORSYTH, D. Retexturing single views using texture and shading. In *European Conference on Computer Vision* (2006), vol. LNCS 3954, pp. 70–81.

[174] WOODHAM, R. Photometric method for determining surface orientation from multiple images. *Optical Engineering 19*, 1 (January 1980), 139–144.

[175] WU, X., DOWNES, M. S., GOKTEKIN, T., AND TENDICK, F. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum* (2001), pp. 349–358.

[176] XIAO, J., BAKER, S., MATTHEWS, I., AND KANADE, T. Real-time combined 2d+3d active appearance models. In *Conference on Computer Vision and Pattern Recognition* (2004), pp. 535–542.

[177] XIAO, J., CHAI, J.-X., AND KANADE, T. A closed-form solution to non-rigid shape and motion recovery. In *European Conference on Computer Vision* (2004), pp. 573–587.

[178] XIAO, J., AND KANADE, T. Uncalibrated perspective reconstruction of deformable structures. In *International Conference on Computer Vision* (2005).

[179] YUILLE, A., COUGHLAN, J. M., AND KONISHI, S. The kgbr viewpoint-lighting ambiguity. *Journal of the Optical Society of America A 20*, 1 (2003), 24–31.

[180] ZHANG, R., TSAI, P.-S., CRYER, J. E., AND SHAH, M. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 8 (1999), 690–706.

[181] ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. Large mesh deformation using the volumetric graph laplacian. *ACM SIGGRAPH 24*, 3 (2005), 496–503.

[182] ZHU, J., AND LYU, M. R. Progressive finit newton approach to real-time nonrigid surface detection. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).

[183] ZIENKIEWICZ, O. *The Finite Element Method.* McGraw-Hill, 1989.

[184] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive multiresolution mesh editing. In *ACM SIGGRAPH* (1997), pp. 259–268.

Curriculum Vitae

Mathieu Salzmann
Born April 12, 1979, Switzerland
E-mail: mathieu.salzmann@a3.epfl.ch

Education

2004-2008 Ph.D. candidate in Computer Vision. Advisor: Prof. Pascal Fua.
Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL)
Title: Learning and Recovering 3D Surface Deformations

1999-2004 M.S. and B.S. in Computer Science
École Polytechnique Fédérale de Lausanne (EPFL)

2001-2002 Exchange Student at Iowa State University (ISU)

1998-1999 Special Mathematics Course (CMS)
École Polytechnique Fédérale de Lausanne (EPFL)

1995-1998 Swiss Professional Technical Maturity (MPT)
and Certificate of Capacity (CFC) in Electronics
École Technique et des Métiers de Lausanne (ETML)

Teaching Experience

2007-2008 Teaching Assistant for Introduction to Computer Vision.
Instructor: Prof. Pascal Fua

2007 Project Supervision. Student: Amin Mazloumian
Title: Implementation of a Real-Time Fluid Simulator.

2005-2006 Teaching Assistant for C++ classes.
Instructor: Dr. Francois Fleuret

Awards

2004 Second best average grade of the Computer Science Departement
École Polytechnique Fédérale de Lausanne (EPFL)

1995-1998 Best average grade for the Swiss Professional Technical Maturity
École Technique et des Métiers de Lausanne (ETML)

Journal Publications

1. Ilić, S., Salzmann, M., and Fua, P. Implicit Meshes for Effective Silhouette Handling. *International Journal of Computer Vision 72*, 2 (February 2007), 159–178.
2. Salzmann, M., Pilet, J., Ilić, S., and Fua, P. Surface Deformation Models for Non-Rigid 3–D Shape Recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 8 (August 2007), 1481–1487.

Conference Publications

1. 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).
2. Salzmann, M., Ilić, S., and Fua, P. Physically Valid Shape Parameterization for Monocular 3–D Deformable Surface Tracking. In *British Machine Vision Conference* (Oxford, UK, September 2005).
3. Salzmann, M., Lepetit, V., and Fua, P. Deformable Surface Tracking Ambiguities. In *Conference on Computer Vision and Pattern Recognition* (Minneapolis, MI, June 2007).
4. Salzmann, M., Hartley, R., and Fua, P. Convex Optimization for Deformable Surface 3–D Tracking. In *International Conference on Computer Vision* (Rio, Brazil, October 2007).
5. Salzmann, M., Urtasun, R., and Fua, P. Local Deformation Models for Monocular 3D Shape Recovery. In *Conference on Computer Vision and Pattern Recognition* (Anchorage, Alaska, June 2008).
6. Lagger, P., Salzmann, M., Lepetit, V., and Fua, P. 3D Pose Refinement from Reflections. In *Conference on Computer Vision and Pattern Recognition* (Anchorage, Alaska, June 2008).
7. Salzmann, M., Moreno-Noguer, F., Lepetit, V., and Fua, P. Closed-Form Solution to Non-Rigid 3D Surface Registration. In *European Conference on Computer Vision* (Marseille, France, October 2008).