Implicit Surfaces Make for Better Silhouettes

Computer Vision Laboratory
Swiss Federal Institute of Technology
CVLAB-IC-BC,1015 Lausanne, Switzerland

Technical Report No: IC/2004/92*

4th November 2004

Abstract

This paper advocates an implicit-surface representation of generic 3–D surfaces to take advantage of occluding edges in a very robust way. This lets us exploit silhouette constraints in uncontrolled environments that may involve occlusions and changing or cluttered backgrounds, which limit the applicability of most silhouette based methods.

This desirable behavior is completely independent from the way the surface deformations are parametrized. To show this, we demonstrate our technique in three very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations; reconstructing the shape of a human face parametrized in terms of a Principal Component Analysis model.

1 Introduction

Occluding contours are a key clue to recovering the shape of smooth and potentially deformable surfaces in monocular sequences and they have been used extensively for this purpose. However, because extracting them reliably against potentially cluttered or changing backgrounds such as those of Fig. 1, is difficult, most of the published work involves engineering the environment to make this task easier.

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

More specifically, we use *implicit meshes* [9], which are implicit surfaces that closely approximate generic triangular 3-D meshes and deform in tandem with them. This formulation allows us to robustly detect the occluding contours on the 3-D surface as the solution of an ordinary differential equation [13]. Their projections can then be used to search for the true image boundaries and deform the 3–D model so that it projects correctly.

This well-formalized approach yields a robust implementation that we demonstrate for monocular tracking of deformable 3–D objects in a completely automated fashion: We start with a generic 3-D model of the target object, find its occluding contours, and use them to search for the corresponding contours in the images. We then use the detected 2-D contours and the constraints they impose, along with some feature information when available, to deform the model.

This approach is effective independently of the specific way the deformations are parametrized. As shown in Fig.1, we validated the tracker in several very different cases: Modeling the deformations of a piece of paper represented by an ordinary triangulated mesh; tracking a person's shoulders whose deformations are expressed in terms of Dirichlet Free Form Deformations [12]; reconstructing the shape of a human face parametrized in terms of a Principal Component Analysis model [1, 7].

In the remainder of the paper, we first review related approaches and our earlier work [9] on implicit meshes. We then show how we use them first to guide the search for silhouettes in the images, and second to enforce the corresponding differential constraints on the surface. Finally, we discuss our results in more details.

2 Related Work

Occluding contours have long been known to be an excellent source of information for surface reconstruction, and

*This work was supported in part by the Swiss National Science Foundation.

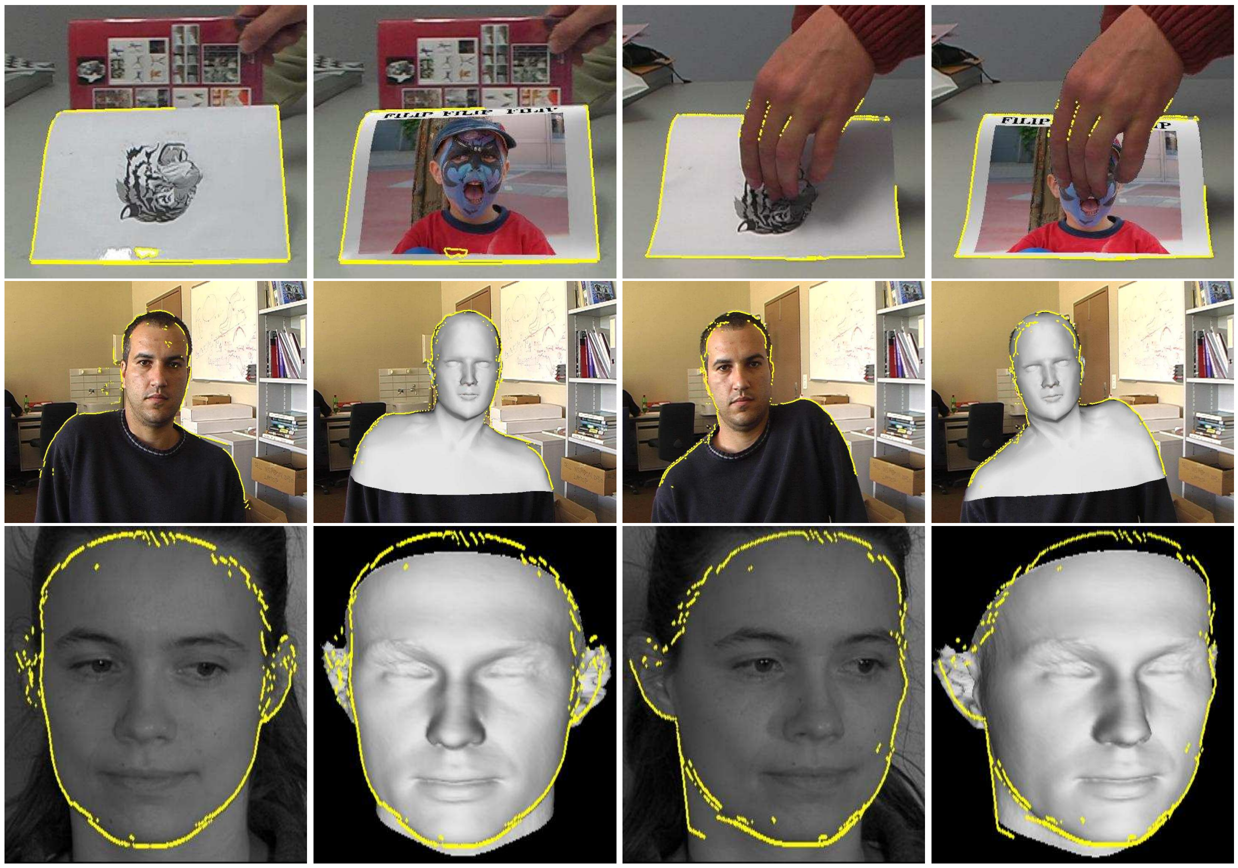


Figure 1: Detecting and using silhouettes for tracking and reconstruction from monocular sequences. The detected silhouette points are shown in yellow, or white if printed in black and white. First row: Tracking a deforming piece of paper with a tiger on it and replacing the tiger by a picture, which involves accurate 3–D shape estimation. This is done in spite of the moving book and the occluding hand. Middle row: Tracking the head and shoulders of a moving person. The reprojected 3–D model is shown as a shaded surface. Note that, even though the background is cluttered, we did not need to perform any kind of background subtraction. Bottom row: Precise reconstruction of a face from a short sequence in which the subject faces the camera.

sometimes the only available one when the surface slants away from the camera and makes it impractical to use other approaches such as feature point matching. This information has been put to very good effect by many researchers, including [16, 20, 6, 17, 2, 5, 18] among many others. In many of these works, the technique used to actually extract the occluding contours is often fairly straightforward. It can be simple edge detection and linking [3], active contour models [10], or space carving [11]. However, while perfectly appropriate in the context in which they are used, these methods would fail in the presence of cluttered and changing backgrounds.

Detecting occluding contours in such situations requires much more sophisticated algorithms. Recent color and texture-based segmentation algorithms [14, 4] have proved very good at this. However, since they are essentially 2–D, it is not trivial to guarantee that the outlines they produce actually correspond to the target object's occluding contours.

A popular solution to this problem among researchers involved in tracking articulated or rigid objects is to model them using volumetric primitives whose occluding contours can be computed given a pose estimate [8, 15]. These contours are then used to search for the true image boundaries in directions that are normal to them. This is effective but has only been demonstrated for relatively simple shapes such as ellipsoids and truncated cones. The work we present here can be understood as a generalization of this approach to more complex surfaces that can deform in less predictable ways.

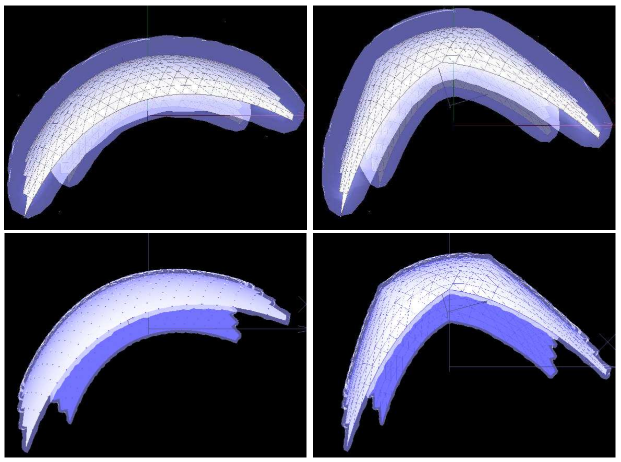


Figure 2: Approximating an explicit mesh by an implicit one. Top row: Spherical implicit meshes wrapped around an explicit mesh and shown as being transparent. Bottom row: Triangular implicit meshes. Note the much improved approximation.

3 Implicit Meshes

In earlier work, we introduced *implicit meshes* [9]. They are implicit surfaces that are designed to closely approximate the shape of arbitrary triangulated meshes and to deform in tandem with them, as shown in Fig. 2. To convert a triangulated mesh into an implicit one, we attach a spherical or triangular implicit surface primitive, and corresponding field function f, to each facet. We then define the surface as the set

$$S(\mathbf{\Theta}) = \{\mathbf{x} \in R^3 \ , F(\mathbf{x}, \mathbf{\Theta}) = T\} \ , \qquad (1)$$

where $F = \sum f_i$, $i = 1..N$ is the sum of the individual field functions, one for each of the N mesh facets, Θ a set of parameters or *state vector* that controls the shape of the explicit mesh, and T a fixed isovalue.

A spherical primitive is created by circumscribing a sphere around the facet i so that the centers of the sphere and of the circle circumscribed around the facet coincide. In this case, f_i simply is

$$f_i(\mathbf{x}) = exp(-k(r_i(\mathbf{x}) - r_i^0)) \ \ i = 1..N, \qquad (2)$$

where $\mathbf{x}$ is a 3–D point, r_i is the Euclidean distance to the sphere's center, $r_i^0$is the radius of the spherical primitive and k is a free coefficient defining slope of the potential field function. For triangular primitives, we replace the Euclidean distance r_i by a piecewise polynomial C^2 function d_i that more accurately approximates the actual distance to the facet i. d_i is computed as the distance from the facet plane for points that project on the facet and as the distance from its edges otherwise. f_i becomes

$$f_i(\mathbf{x}) = exp(-k(d_i(\mathbf{x}) - d_0)) \ \ , \qquad (3)$$

which has almost the same form as before, but where d_0 now represents the thickness of the implicit surface and is the same for all facets.

Spherical primitives are best for relatively regular meshes because they are computationally inexpensive. Triangular primitives are more expensive but also more general and provide better surface approximations, especially when the explicit mesh is either irregular or low resolution. In any event, the method proposed in this paper is applicable to both since it only depends on the surface differentiability.

4 Silhouette Detection

As discussed earlier, given the estimated shape and pose of a 3–D model, our goal is to compute its 3–D occluding contours, project them into the image and use that projection as a starting guess to find the corresponding image boundaries, which should be the real silhouettes. In this section, we first show some of the problems involved in performing this task using traditional techniques. We then show that our implicit mesh formalism solves them and gives us cleaner and more consistent results, which can then be exploited to detect the right image boundaries.

4.1 Occluding Contours from Explicit Meshes

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

As shown in Fig. 3(b,c), the results of this procedure are heavily dependent on mesh resolution and the resulting contours are rarely as smooth as they should. Of course, more sophisticated heuristics would certainly yield improved results but we are not aware of any existing technique whose results are as clean and mesh-resolution independent as those of Fig. 3(d,e), which were obtained using our implicit surface formalism.

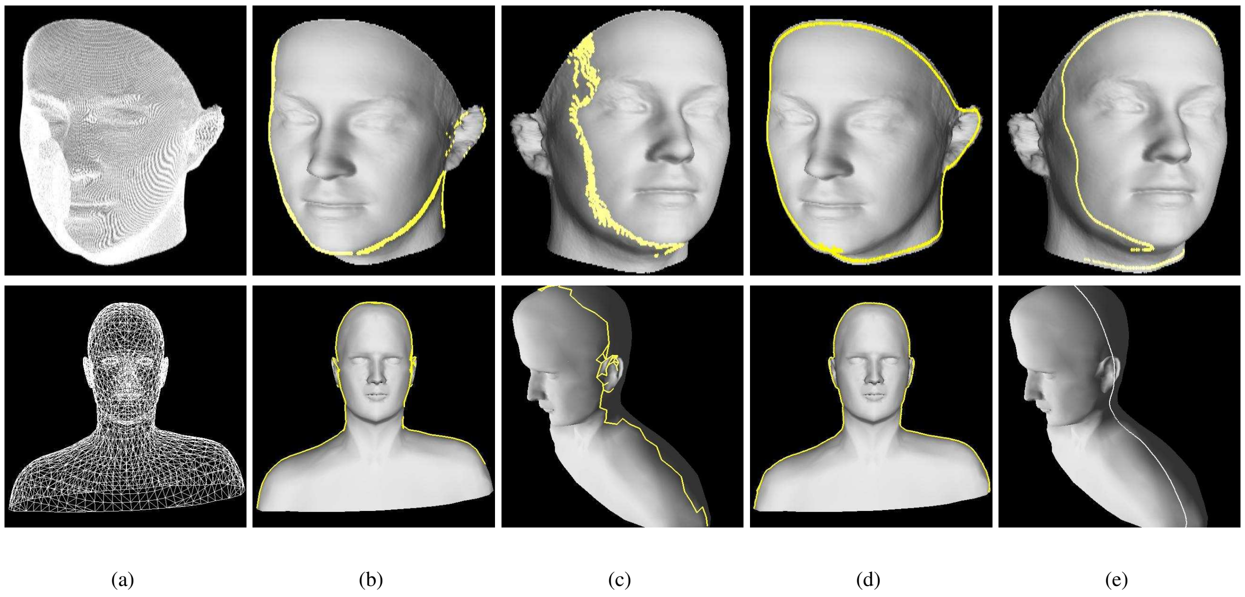


Figure 3: Occluding contours on explicit versus implicit meshes. (a) High resolution mesh of the face and low resolution mesh of the upper body. (b) Shaded model with eges at the boundary between visible and hidden facets overlaid in yellow. (c) The same edges seen from a different viewpoint (d,e) Shaded models with the occluding contour computed using implicit mesh, corresponding to views (b) and (c) respectively. Note the much greater smoothness and improved precision.

4.2 Occluding Contours and Ordinary Differential Equations

As shown in [13], occluding contours of implicit surfaces can be found by solving an ordinary differential equation (ODE) as follows: Let $\mathbf{x}(t)$, $t \in [0,1]$ be a 3–D occluding contour on the implicit surface $S(\mathbf{\Theta})$ of Eq. 1, such as the one depicted by Fig. 4. For all values of t,

1. $\mathbf{x}(t)$ is on the surface and therefore $F(\mathbf{x}(t), \mathbf{\Theta}) = T$,
2. the line of sight is tangential to the surface at $\mathbf{x}(t)$.

This implies [13] that $\mathbf{x}(t)$, $t \in [0,1]$ is a solution of the ODE

$$\frac{\partial \mathbf{x}(t)}{\partial t} = \frac{(H(\mathbf{x}(t))(\mathbf{x}(t) - \mathbf{COpt})) \times \bigtriangledown F(\mathbf{x}(t), \mathbf{\Theta})}{\|(H(\mathbf{x}(t))(\mathbf{x}(t) - \mathbf{COpt})) \times \bigtriangledown F(\mathbf{x}(t), \mathbf{\Theta})\|} \quad (4)$$

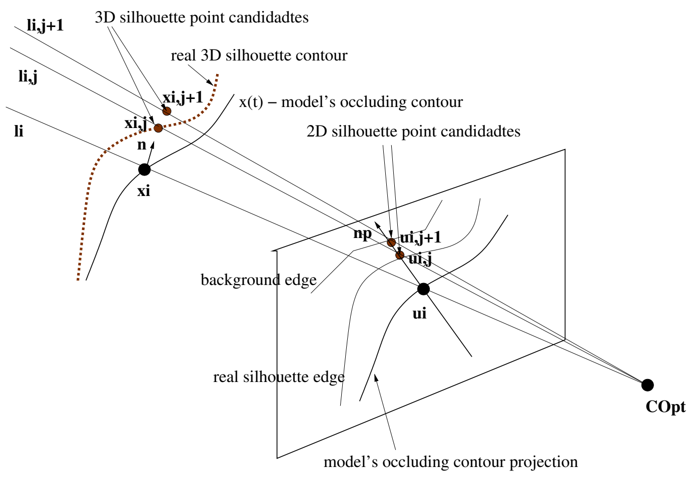
where $H(\mathbf{x}(t))$ is the Hessian matrix of F, $\bigtriangledown F(\mathbf{x}(t))$ its gradient vector and $\mathbf{COpt}$ the optical center of the camera, as shown in Fig. 4.

Solving this ODE requires an appropriate starting point $\mathbf{x}(0)$, that is one 3–D point on the occluding contour. To find one *single* vertex of the explicit mesh that is very likely to be an occluding vertex, we use a visibility algorithm similar to the one described in Section 4.1. We then project it onto the implicit mesh and search in the neighborhood of the projection for a point that satisfies the two above stated constraints. Note that this is very different from the approach of Section 4.1 because, since we only need one 3–D point, we can impose very tight constraints and thus ensure that it really is on the occluding contour. This results in the very clean contours of Fig. 3(d,e) that are quite insensitive to the resolution of the mesh used to compute them.

4.3 Finding Silhouette Edges in the Image

Given a 3–D occluding contour $\mathbf{x}(t)$ computed as described above, we project it into the image and look for the true silhouette edge in a direction normal to its 2–D projection as depicted in Fig. 4. This is geometrically consistent because, at a silhouette point $\mathbf{x}_i \in \mathbf{x}(t)$, $t \in [0,1]$, the 3–D surface normal $\mathbf{n}$ is perpendicular to the line of sight $\mathbf{l}_i$ and, as a result, projects to the normal $\mathbf{n}_p$ of the 2–D contour.

In other words, at each point $\mathbf{u}_i$ of the 2–D projection, we simply have to perform a 1–D search along a scan-line for the true edge location and we are back to the old edge detection problem, but in a much simpler context than usual. We use a technique that has proved effective for edge-based tracking [19, 8]: Instead of selecting one arbitrary gradient maximum along the scan-line, we select multiple gradient maxima resulting in several potential silhouette edge points $\mathbf{u}_i^j$ and corresponding lines of sight $\mathbf{l}_i^j$ for each $\mathbf{x}_i$. Along these new lines of sight, we could choose the $\mathbf{x}_i^j$ where the line is closest to the surface as the most likely



li,j+1
3D silhouette point candidadtes
real 3D silhouette contour
li,j
xi,j+1
x(t) – model's occluding contour
xi,j
2D silhouette point candidadtes
li
n
xi
np
ui,j+1
ui,j
background edge
ui
real silhouette edge
COpt
model's occluding contour projection

Figure 4: Finding multiple silhouette edge points in the image. Notations are defined in Section 4.3

point where the surface should be tangent to the line of sight. However, this involves a computationally expensive search along the line of sight. In practice, as shown in Fig. 4, a simpler and equally effective approach is to take each $\mathbf{x}_i^j$ to be the point on $\mathbf{l}_i^j$ that is at the same distance from the optical center as the original $\mathbf{x}_i$. These $\mathbf{x}_i^j$ are then used as silhouette observations, as explained in Section 5.

5 Fitting Implicit Mesh 3–D Models

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

In this section, we show that our formalism allows us to effectively combine these two information sources. Given a set of correspondences and silhouette points, we fit our model to the data by minimizing a set of *observation equations* in the least-squares sense. To this end we use the Levenberg-Marquardt algorithm and, at each iteration, we recompute the occluding contours and corresponding silhouette points in the image using the technique of Section 4.

As we will see, the silhouette-based constraints are best expressed in terms of the implicit surface formalism while it is simpler to formulate the correspondence-based ones using traditional triangulated meshes. Recall from Section 3 that both the implicit mesh and the underlying explicit one deform in tandem when the state vector changes. As a result, we can simultaneously use the implicit formalism when dealing with silhouettes and the explicit one when dealing with correspondences as needed to simplify our implementation. We view this as one of the major strengths of our approach.

5.1 Least Squares Framework

We use the image data to write n_{obs} observation equations of the form

$$Obs(\mathbf{x}_i, \mathbf{\Theta}) = \epsilon_i \ , \ 1 \le i \le n_{\text{obs}} \ , \tag{5}$$

where $\mathbf{x}_i$ is a data point, $\mathbf{\Theta}$ the state vector of Eq. 1, Obs a differentiable function whose value is zero for the correct value of $\mathbf{\Theta}$ and completely noise free data, and ϵ_i is treated as an independently distributed Gaussian error term. We then minimize $v^T P v$, where $v = [\epsilon_1, \ldots, \epsilon_{n_{\text{obs}}}]$ is the vector of residuals and P is a diagonal weight matrix associated with the observations. Our system must be able to deal with observations coming from different sources, here occluding contours and point correspondences, that may not be commensurate with each other. We therefore associate to each data point $\mathbf{x}_i$ an observation type $type_i$ and to each type a weight w^{type}chosen so that the contribution to the objective function gradients of all the observations of a particular kind are of similar magnitudes [9].

Because there are both noise and potential gaps in the image data, we add a regularization term E_D that forces the deformations to remain smooth and whose exact formulation depends on the kind of model we use. The total energy that we minimize therefore becomes:

$$E_T = \sum_{i=1}^{nobs} w^{type_i} \left\| Obs^{type_i}(\mathbf{x}_i, \Theta) \right\|^2 + E_D, \tag{6}$$

where Obs^{type} is the function that corresponds to a particular observation type. We now turn to the description of these functions for the two data types we use.

5.2 Silhouettes

In Section 4, we showed how to use our formalism to associate 2–D image locations to 3–D surface points that lie on the occluding contours. If the shape and pose of the 3–D model were perfect, the 3–D points would project exactly at those locations. In other words, for each i, at least one of the candidate occluding points $\mathbf{x}_i^j$ introduced at the end of Section 4.3 should be on the surface, as shown in Fig. 4. During the optimization, this will in general not be true and we enforce this constraint by introducing a *silhouette function* of the form

$$Obs^{silh}(\mathbf{x}_i^j, \mathbf{\Theta}) = w_i^j (F(\mathbf{x}_i^j, \mathbf{\Theta}) - T), \tag{7}$$

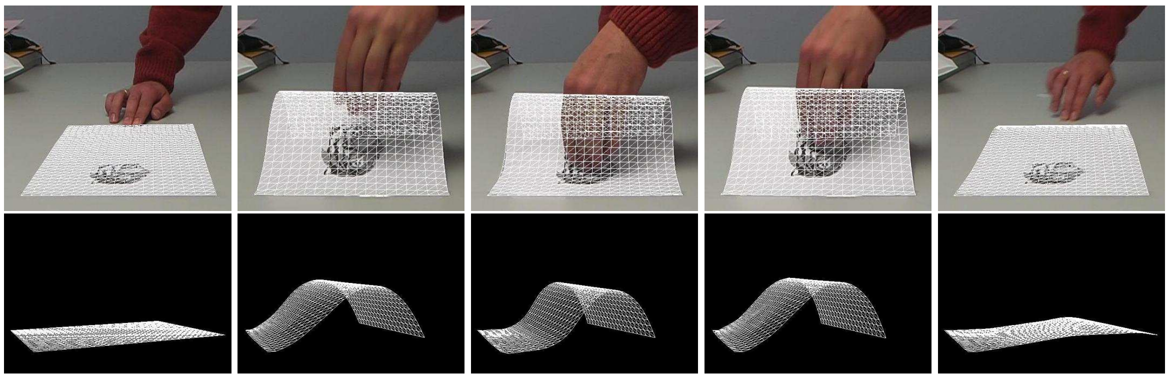


Figure 5: Occlusion handling. The front of the paper is taped to the table and one hand pushes the back of the page while the other passes in front. Top row: The recovered mesh is overlaid on the images. Note that the hand is *in front* of the paper even though the wireframed display gives the impression that it is behind. Bottom row: Side view of the recovered mesh. Note that its shape is undisturbed by the occlusion and that the back of the mesh also deforms correctly. A video of the sequence is given as supplementary material.

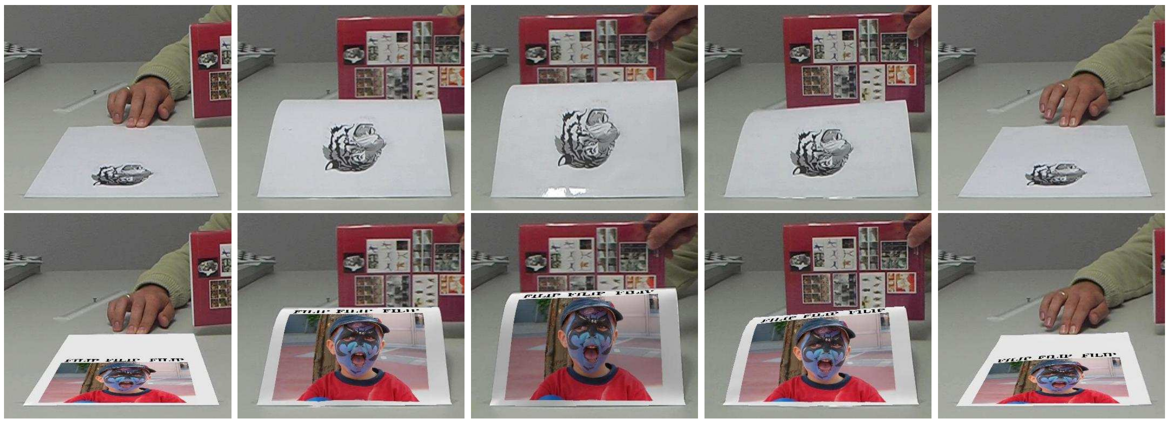


Figure 6: Handling a changing background. Top row: Original sequence with book sliding in the background. Bottom row: A new texture is applied on the deformed mesh and reprojected in the images. Note that background subtraction techniques could not have been applied in this case. A video of the sequence is given as additional material.

for each $\mathbf{x}_i^j$, where w_i^j is the weight associated to the candidate, F the field function of Eq. 1, and T the isovalue defined in the same equation.

For each $\mathbf{x}_i^j$, w_i^j is taken to be inversely proportional to its distance to the line of sight $\mathbf{l}_i$. As a result, for each i only one of these candidates, $\mathbf{x}_i^{best}$, will end up being on $\mathbf{l}_i$ while the others will eventually be ignored. As the total energy E_T of Eq. 6 is minimized, the $Obs^{silh}(\mathbf{x}_i^j, \mathbf{\Theta})$ will collectively decrease in the least-squares sense and x_i^{best} will become closer and closer to actually being on the surface. Note that, because $\mathbf{x}_i^{best}$ minimizes the distance to the surface along the corresponding line of sight, the normal to the closest surface point is perpendicular to it. Thus, $\mathbf{x}_i^{best}$ will eventually tend to satisfy the two conditions that characterize a point on an occluding contour introduced in Section 4.2.

5.3 Correspondences

We use 2–D point correspondences in pairs of consecutive images as our additional source of information: We find interest points in the first image of the pair and establish correspondences in the second using a simple correlation-based algorithm. Given a couple $\mathbf{u}_i = (p_i^1, p_i^2)$ of corresponding points found in this manner , we define a *correspondence function* $Obs^{corr}(\mathbf{u}_i, \Theta)$ as follows: We back-project p_i^1 to the 3–D surface and reproject it to the second image. We then take $Obs^{corr}(\mathbf{u}_i, \Theta)$ to be the Euclidean distance in the image plane between this reprojection and p_i^2.

Note that the simplest and fastest way of backprojecting p_i^1 to the surface is to use OpenGl and the graphics hardware of our machines to find the facet that is traversed by the line of sight defined by p_i^1. Therefore in our implementation, when computing $Obs^{corr}(\mathbf{u}_i, \Theta)$ and its derivatives, we use the explicit representation instead of the implicit one.

6 Results

In previous sections, we claimed that our formalism applies independently of the specific parametrization used to represent the deformations. Here we demonstrate this in three different cases.

6.1 Tracking a Piece of Paper

We model the paper as a rectangular mesh parametrized in terms of the coordinates of its vertices. To keep the deformations physically plausible, we define a deformation energy that is the sum of two terms. The first one represents the inextensibility of the paper by penalizing the variations of the distance between a vertex and its neighbors. The second one models the bending stiffness of paper by constraining the curvature of the mesh.

Fig. 5 shows the results obtained when the paper is partially occluded. The first row shows the deformed mesh we obtain overlaid as a white wireframe on the original images. The second row shows the side view of the same deformed mesh. We can see that the back of the mesh also deforms in a coherent manner. Even though the silhouette contours are partially hidden, our algorithm still retrieves the correct deformation and keeps on tracking the piece of paper. In the two upper right images of Fig. 1, we used a different image as a texture map to replace the tiger by a boy's face. To avoid hiding the hand, we texture-mapped it on a plane that is closer than the mesh.

Fig. 6 highlights the robustness of our algorithm to a changing background. The first row shows the original sequence with the same tiger image as before and a moving book behind. In the second row, we used the deformed mesh to map a new texture onto the images. The new images look realistic and such results couldn't have been obtained by using a simple background substraction technique.

6.2 Head and Shoulders Tracking

Here we apply our method to recovering the motion of moving head and shoulders in monocular sequences. Since the meshes used here are of much higher resolution than before, parametrizing them in terms of the vertices coordinates would have been computationally too expensive. Therefore we used a Dirichlet Free Form Deformations (DFFD) parametrization so that the shape of our model depends only on a small set of DFFD control points that form a *control mesh* [12]. In order to enforce a smooth deformation, neighboring vertices of the control mesh must deform in a relatively similar manner. This is achieved by using a deformation energy that approximates the sum of the square of the derivatives across the control surface [9].

For each subject, we first build a 3–D model from a sequence where the person does not move but the camera does. This model is then used to track the motion in sequences such as the one in the first row of Fig. 7. In this case, interest points are found on the head while occluding contours are used for the neck and shoulders. This results in the reconstruction of Fig. 1. As shown in the second row of Fig. 7, it can be used to resynthetize the subject in front of a different background, thus eliminating the need for a blue screen.

6.3 Head Modeling

In earlier work [7], we have shown that we can recover the shape and camera motion from uncalibrated sequences using a PCA based face model [1] and 2–D image correspondences. Here, we extend this approach by also incorporating occluding contour information. The deformation energy now penalizes PCA parameters values that are too far from acceptable values for faces [1].

In the third row of Fig. 1 we reproject the recovered face model into the images. Note that the 2–D silhouettes found in the images perfectly match the projected model outlines.

For comparison's sake, in Fig. 8 we show side by side the results obtained using correspondences alone and those obtained by adding the silhouettes. Note that the latter are noticeably improved.

7 Conclusion

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

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

In other words, our implicit surface based approach to using silhouettes is appropriate for uncontrolled environments that may involve occlusions and changing or clut-

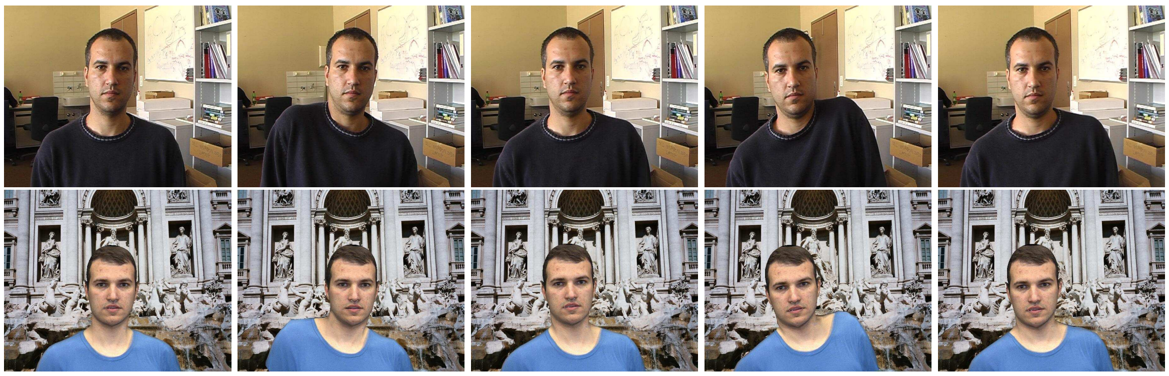


Figure 7: Tracking of moving head and shoulders. The model was first reconstructed from an uncalibrated video sequence, not shown here but given as supplementary material, from which the texture was taken. Top row: Original sequence used to track the person. Bottom row: Recovered model placed in front of a different background. A video given as supplementary material shows all these sequences.

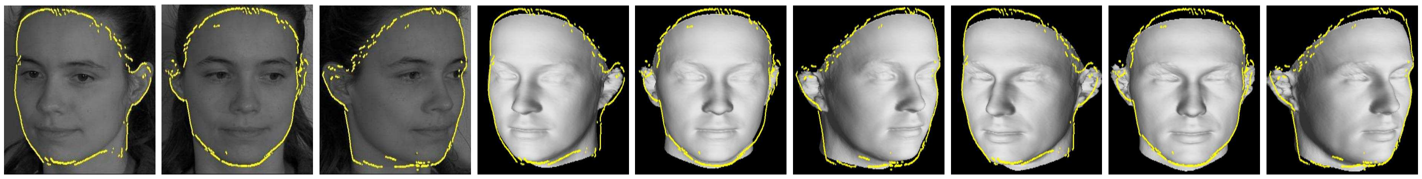


Figure 8: Head modeling using PCA face models. (a) Three images from a short video sequence with image silhouette edges detected by using our technique. (b) Recovered face shape using only interest points with the same silhouettes as before. Note that they do not match exactly. (c) Recovered shape using both silhouettes and interest points. The occluding contours of the model now corresponds almost exactly to the silhouette edges. A short video showing additional views is given as supplementary material.

tered backgrounds, which limit the applicability of most other silhouette-based methods. Furthermore, our approach is independent from the way the surface deformations are parametrized, as long as this parameterization remains differentiable.

References

[1] V. Blanz and T. Vetter. A Morphable Model for The Synthesis of 3–D Faces. In *Computer Graphics, SIGGRAPH Proceedings*, Los Angeles, CA, August 1999.

[2] E. Boyer and M.-O. Berger. 3D Surface Reconstruction Using Occluding Contours. *International Journal of Computer Vision*, 22(3):219.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[4] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026, 2002.

[5] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, 9(2):83–112, 1992.

[6] G. Cross and A. Zisserman. Surface reconstruction from multiple views using apparent contours and surface texture. In A. Leonardis, F. Solina, and R. Bajcsy, editors, *NATO Advanced Research Workshop on Confluence of Computer Vision and Computer Graphics, Ljubljana, Slovenia*, pages 25–47, 2000.

[7] M. Dimitrijevic, S. Ilic, and P. Fua. Accurate Face Models from Uncalibrated and Ill-Lit Video Sequences. In *Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004.

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

[9] S. Ilic and P. Fua. Generic Deformable Implicit Mesh Models for Automated Reconstruction. In *ICCV Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, Nice, France, October 2003.

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[11] K.N. Kutulakos and S.M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3):197–216, July 2000.

[12] L. Moccozet and N. Magnenat-Thalmann. Dirichlet Free-Form Deformation and their Application to Hand Simulation. In *Computer Animation*, 1997.

[13] E. Rosten and Tom Drummond. Rapid rendering of apparent contours of implicit surfaces for realtime tracking. In *British Machine Vision Conference*, volume 2, pages 719–728, Norwich, UK, 2003.

[14] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[15] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *Conference on Computer Vision and Pattern Recognition*, volume I, Madison, WI, June 2003.

[16] S. Sullivan, L. Sandford, and J. Ponce. Using geometric distance fits for 3–d. object modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1183–1196, December 1994.

[17] R. Szeliski and R. Weiss. Robust Shape Recovery from Occluding Contours Using a Linear Smoother. *International Journal of Computer Vision*, 28(1):27.

[18] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:703–714, 1991.

[19] L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. Arlington, VA, November 2004.

[20] R. Vaillant and O.D. Faugeras. Using extremal boundaries for 3D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):157 – 173, 1992.