Pattern Recognition Letters 30 (2009) 1198–1207

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Shape from silhouette using topology-adaptive mesh deformation ☆

Y. Yemez *, Y. Sahillioğlu

Multimedia, Vision and Graphics Laboratory, College of Engineering, Koç University, Sarıyer, Istanbul 34450, Turkey

ARTICLE INFO

Article history:
Received 20 November 2008
Received in revised form 22 May 2009
Available online 21 June 2009

Communicated by M. Couprie

Keywords:
Shape from silhouette
Surface reconstruction
Mesh deformation
Robust topology control

ABSTRACT

We present a computationally efficient and robust shape from silhouette method based on topology-adaptive mesh deformation, which can produce accurate, smooth, and topologically consistent 3D mesh models of complex real objects. The deformation scheme is based on the conventional snake model coupled with local mesh transform operations that control the resolution and uniformity of the deformable mesh. Based on minimum and maximum edge length constraints imposed on the mesh, we describe a fast collision detection method which is crucial for computational efficiency of the reconstruction process. The topology of the deformable mesh, which is initially zero genus, can be modified whenever necessary by merging operations in a controlled and robust manner by exploiting the topology information available in the silhouette images. The performance of the proposed shape from silhouette technique is demonstrated on several real objects.

1. Introduction

The shape from silhouette technique has recently regained interest since its robust output forms a solid initial model for further tasks of computer vision. Early examples of this technique were presented in (Chien and Aggarwal, 1986; Szeliski, 1993) and later much improvement has been established concerning accuracy and efficiency issues (Niem and Wingbermuhle, 1999; Tarini et al., 2002; Boyer and Franco, 2003; Fang et al., 2003; Yemez and Schmitt, 2004; Erol et al., 2005). Silhouette models cannot capture hidden surface concavities, but can further be carved or deformed so as to achieve a more accurate object representation by incorporating stereo or optical triangulation information (Esteban and Schmitt, 2004; Yemez and Wetherilt, 2007). Moreover, the shape from silhouette technique, as a passive reconstruction method, can successfully be used to model time-varying scenes and objects (Mueller et al., 2004; Carranza et al., 2003; Bilir and Yemez, 2008) whereas most of the active scene capture technologies become inapplicable in the dynamic case.

Shape from silhouette techniques existing in the literature commonly make use of an intermediate volumetric representation, e.g., in terms of cubic voxels or tetrahedra, which is then converted to a surface mesh representation by means of a triangulation method such as the marching cubes algorithm (Newman and Yi, 2006) or Delaunay triangulation (Boyer and Franco, 2003). These techniques can be computationally very efficient and accurate, and can generate topologically consistent (i.e., manifold) mesh representations. However, triangulation methods that they use rely only on local surface information and hence they always have to deal with topological ambiguities. Adhering to very high resolution representations, which can be obtained in an efficient manner by adaptive sampling strategies as in (Erol et al., 2005), may alleviate but cannot totally eliminate the topological ambiguity problem. Since these techniques do not have an explicit or global control on topology, they can easily yield representations which are topologically incorrect, i.e., which do not have the same topology as the object to be reconstructed. Moreover, since volumetric visual hull techniques are commonly based on the idea of space carving with no inherent mechanism of smoothing, their reconstructions, especially at high resolutions, are very sensitive to imperfections of the silhouette extraction and camera calibration processes, and may hence severely suffer from geometric and topological distortions.

Deformable mesh models, which in general yield smooth and robust representations, were successfully used before for the problem of 3D shape recovery in different contexts such as shape from range data, shape from stereo and segmentation of volumetric images (Terzopoulos et al., 1991; Park et al., 2001; Lachaud and Taton, 2003; Esteban and Schmitt, 2004; Duan et al., 2004). However, the use of mesh deformation, which we advocate in this paper, has not been considered so far for the shape from silhouette problem. The reason seems to be twofold. First, mesh deformation methods are considered as computationally inefficient as compared to visual hull techniques based on space carving. Second and more

☆ This work has been supported by TUBITAK under the project EEEAG-105E143 and by the European Network of Excellence 3DTV (www.3dtv-research.org).

* Corresponding author. Tel.: +90 212 3381585; fax: +90 212 3381548.
E-mail addresses: yyemez@ku.edu.tr (Y. Yemez), ysahillioglu@ku.edu.tr (Y. Sahillioğlu).

0167-8655/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2009.05.012

importantly, it is a challenging problem for deformable mesh models to recover the shape of an object with arbitrary genus based on only silhouette information. The primary contribution of this paper is a shape from silhouette technique based on mesh deformation, which addresses these two challenges.

The deformable model that we employ is based on the deformation scheme proposed in (Lachaud and Taton, 2003) for 3D segmentation of complex anatomical structures. In this scheme, an initial mesh model is iteratively deformed towards the target boundary by external and internal forces. The resolution and structural uniformity of the deformable mesh are controlled during surface evolution via edge collapse, split and flip operations. The most problematic issue in this method, as also in most mesh-based deformation schemes, concerns the topology control which enables recovery of the shapes with arbitrary genus. The common practice is to incorporate topology modifying operators (splitting and merging) into the deformation scheme, such as in (Lachaud and Taton, 2003and Duan et al., 2004), whenever a collision is detected between distant parts of the deformable surface. However, this strategy relies on a false assumption which may easily lead to incorrect and/or redundant topology modifications. Collisions may indeed occur due to the limited flexibility of the deformable model as well as due to a need for topology change. The deformable model may not be fast or flexible enough to penetrate into deep concavities at a given resolution and even partially get stuck at some part of the surface at some iteration of the evolution while distant surface parts of the model keep getting close to each other, yielding unexpected collisions. The challenge is to be able to correctly decide whether a detected collision requires modification of the topology or it is simply a misguided self-intersection of the deformable model.

In view of the above discussion, the contributions of this paper can be summarized as follows:

- We adopt the deformable mesh described in (Lachaud and Taton, 2003) to address the shape from silhouette problem. We define an isolevel function based on bilinear interpolation of the silhouette images to be used as the external force of the deformable model along with a fine-tuning strategy that accurately places the model onto the boundary surface. This isolevel function is also used to assess the geometrical accuracy of the deformable mesh to support adaptive resolution. The resulting reconstructions are accurate and always smooth even when the silhouettes are corrupted with noise.
- We propose a robust topology control mechanism. The reconstruction scheme starts with an initial deformable mesh of genus zero. Topological changes are introduced during deformation whenever necessary in a controlled manner based on the topology of silhouette images. Our scheme also allows an optional user interaction mechanism to improve robustness of the topology control when modeling geometrically very complicated objects. Our method does not guarantee topological correctness for all possible types of surfaces but can ensure that the topology is correctly recovered for most of the typical real-world objects with arbitrary genus.
- We describe a fast and effective method for collision detection and handling, based on the minimum and maximum edge length constraints imposed on the deformable mesh, which is crucial for computational efficiency of the deformation scheme.

In Section 2, we describe the surface deformation method that we employ and then in Section 3, we address the shape from silhouette problem. Our topology control mechanism is described in Section 4 along with the overall reconstruction algorithm. Section 5 presents the experimental results and Section 6 finally provides concluding remarks and perspectives for future research.

2. Deformation algorithm

The deformation method that we use is based on the active contour models, or so called "snakes", which were first developed in (Kass et al., 1988). The method basically follows the Lagrangian approach: An initial deformable mesh S_0 representing the bounding sphere is iteratively evolved towards the boundary of the object under the guidance of internal and external forces that try to minimize an overall energy. This surface evolution can be expressed by the following equation:

$$S_k = S_{k-1} + \boldsymbol{F}_{\text{int}}(S_{k-1}) + \boldsymbol{F}_{\text{ext}}(S_{k-1}), \tag{1}$$

where $\boldsymbol{F}_{\text{int}}$ and $\boldsymbol{F}_{\text{ext}}$ denote the internal and external forces. By iterating the above equation, the surface S_k converges to its optimum at the equilibrium condition when all the forces cancel out to 0. The external force component, $\boldsymbol{F}_{\text{ext}}$, is application-specific and commonly set to be in the direction of the surface normal:

$$\boldsymbol{F}_{\text{ext}}(P) = v(P) \cdot \boldsymbol{N}(P), \tag{2}$$

where $\boldsymbol{N}(P)$ is the normal vector and $v(P)$ is the force strength at vertex P of the deformable mesh.

The internal force component, $\boldsymbol{F}_{\text{int}}$, controls the smoothness of the mesh. We employ a combination of tangential Laplacian operator (Wood et al., 2000) and Taubin's fairing technique (Taubin, 1995), which is easy-to-compute, and yields a smooth deformation with no significant geometrical shrinkage and bias in the final surface estimate. The internal force $\boldsymbol{F}_{\text{int}}(P)$ is defined in terms of its components along two orthogonal directions,

$$\boldsymbol{F}_{\text{int}}(P) = \boldsymbol{F}_{\text{T}}(P) + \boldsymbol{F}_{\text{N}}(P), \tag{3}$$

where the components $\boldsymbol{F}_{\text{T}}$ and $\boldsymbol{F}_{\text{N}}$ correspond to smoothing along tangential and normal directions of the surface, respectively. We obtain the tangential component, $\boldsymbol{F}_{\text{T}}$, by tangential Laplacian smoothing:

$$\boldsymbol{F}_{\text{T}}(P) = \mathscr{L}(P) - (\mathscr{L}(P) \cdot \boldsymbol{N})\boldsymbol{N}, \tag{4}$$

where $\mathscr{L}(P)$ denotes the Laplacian operator that moves a vertex to the centroid of its one-ring neighbors. The component $\boldsymbol{F}_{\text{N}}$ is obtained by fairing the surface along its normal direction:

$$\boldsymbol{F}_{\text{N}}(P) = (\mathscr{F}(P) \cdot \boldsymbol{N})\boldsymbol{N}, \tag{5}$$

where $\mathscr{F}(P)$ is the displacement created by the non-shrinking surface fairing algorithm described in (Taubin, 1995).

Following (Lachaud and Taton, 2003), we incorporate three local mesh restructuring operators (Kobbelt et al., 2000), namely edge collapse, edge split and edge flip, into the surface deformation process, to handle degenerate edges and irregular vertices and to impose uniformity on the mesh structure. At each iteration of the above algorithm, split operations are first applied to the deformable mesh to remove edges with length longer than $\varepsilon_{\max}$. The same process is then repeated with collapse operations to remove edges shorter than $\varepsilon_{\min}$. Next to split and collapse operations, which inevitably change the valence distribution of the mesh structure, edge flip operations are applied by swapping the common edge of any two neighboring triangles with the one joining the unshared vertices of the triangles as long as this operation favors the existence of the vertices of valence close to 6. We set $\varepsilon_{\max} = 2\varepsilon_{\min}$ to have uniformly sized triangles with small aspect ratios as proposed in (Kobbelt et al., 2000). Since the edge length ratio is then bounded by $\varepsilon_{\max}/\varepsilon_{\min} = 2$ and the valence distribution preserves its uniformity by flip operations, the deformable mesh keeps its initial high quality in terms of the aspect ratio of the triangles during surface evolution.

For a stable mesh evolution process, it is important to comply with the minimum and maximum edge length constraints. Unfortunately, it is usually impossible to achieve this via successive

application of the split and collapse operators since edge splits may create edges shorter than $\varepsilon_{\min}$, and likewise edge collapses may create edges longer than $\varepsilon_{\max}$. Iterating split-collapse operators does not alleviate the problem either since some edges may keep alternating between these operations. Hence there is a choice to be made here. Depending on whether the split or collapse operator is first applied in the algorithm, only one of the constraints can be guaranteed. We choose to strictly comply with the minimum edge length constraint and apply the split operator first since the minimum edge length constraint is more important for stability. As a result, some large edges may occasionally appear, but they mostly remain isolated and do not persist long, being usually handled at the next iterations of the algorithm by edge collapses. Similarly, edge flips can also create short edges. In those occasions, we simply do not apply the flip operation to comply with the minimum edge length constraint.

The above strategy alone does not however guarantee the minimum edge length constraint since a collapse operation is not always a legal move, in other words, its application may create non-manifold triangulations in some certain occasions. As explained in (Hoppe et al., 1993), the collapse of an edge defined by two vertices P_i and P_j is legal in a closed manifold mesh if and only if for all vertices P_k adjacent to both P_i and P_j, $\{P_i, P_j, P_k\}$ is a face of the mesh. To strictly comply with the minimum edge length constraint, we follow the following strategy: Whenever an illegal collapse operation is encountered, we first detect those vertices P_k for which $\{P_i, P_j, P_k\}$ is not a face, remove them from the mesh structure, and then safely apply the collapse operation. We note that a flip operation can also be illegal. We detect and do not perform such illegal flip operations that would otherwise yield non-manifold triangulations.

Another source of instability for mesh evolution is self-collisions. Although self-intersection of the deformable mesh is avoided during deformation via a collision detection mechanism as described later in Section 4, it is essential to keep the number of collisions as small as possible and to definitely avoid any collision between neighboring vertices. We constrain the strength of the external force with half of the minimum edge length, i.e., with $|v| \leqslant \varepsilon_{\min}/2$, so that neighboring vertices never interfere with each other and yield self-intersections.

3. Shape from silhouette

3.1. Silhouette-based external force

The external force component, $\boldsymbol{F}_{\text{ext}}$, is computed based on the silhouette information. Recalling Eq. (2), the strength of the external force $v(P)$ at each vertex P of the mesh and at each iteration of the surface evolution is based on how far and in which direction (inside or outside) the vertex P is with respect to the silhouettes. Thus the force strength v, which may take negative values as well, is computed by projecting P onto the image planes and thereby estimating an isolevel value $f(P)$ via bilinear interpolation:

$$v(P) = \varepsilon_{\min} f(P) = \varepsilon_{\min} \min_n \{G[\text{Proj}_{I_n}(P)] - 0.5\}, \tag{6}$$

where $\text{Proj}_{I_n}(P)$ is the projection of the vertex P to I_n, the n'th binary image (0 for outside, 1 for inside) in the sequence, and

$$\begin{aligned} G(x',y') = {} & (1-\alpha)((1-\beta)I(\lfloor x' \rfloor, \lfloor y' \rfloor) + \beta I(\lfloor x' \rfloor + 1, \lfloor y' \rfloor)) \\ & + \alpha((1-\beta)I(\lfloor x' \rfloor, \lfloor y' \rfloor + 1) + \beta I(\lfloor x' \rfloor + 1, \lfloor y' \rfloor + 1)), \end{aligned}$$

where $(\lfloor x' \rfloor, \lfloor y' \rfloor)$ denotes the integer part and (α, β) is the fractional part of the coordinate (x', y') in the binary discrete image I. The function G, taking values between 0 and 1, is the bilinear interpolation of the sub-pixelic projection (x', y') of the vertex P. Thus, the isolevel function $f(P)$ takes values between -0.5 and 0.5, and the zero crossing of this function reveals the isosurface. The isovalue of the vertex P is provided by the image of the silhouette that is farthest away from the point, or in other words, where the interpolation function G assumes its minimum value.

We distinguish the vertices of the deformable mesh under three categories with respect to their isovalues: IN ($f(P) = 0.5$), ON ($-0.5 < f(P) < 0.5$) or OUT ($f(P) = -0.5$). According to this definition, ON vertices are those positioned within a narrow band around the boundary surface. This band has a thickness of approximately two pixels when projected onto the image planes. By Eq. (6), the external force strength at each ON vertex varies within the interval $(-\varepsilon_{\min}/2, \varepsilon_{\min}/2)$. The vertices which are out of this band are labeled as either IN or OUT and have fixed force strength $\varepsilon_{\min}/2$ or $-\varepsilon_{\min}/2$, respectively.

3.2. Fine tuning

During surface evolution, the state of a vertex, which is initially OUT, can switch between any two of the three categories. A vertex moves not only due to the external force, but also due to the regularization effect of the internal force that alters its positioning. Depending on the magnitude of the external force, which is bounded above by $\varepsilon_{\min}/2$, the state of a vertex can even switch from OUT to IN, or vice versa, at one single iteration. The vertices of the deformable mesh, when they get close to the boundary, may oscillate between IN and OUT states until convergence, that is, until they no longer move. Some vertices remain as OUT or IN even at convergence. To improve accuracy and to speed up convergence, we incorporate a fine-tuning procedure to the surface deformation process. We detect the instances when a vertex crosses the target boundary due to the effect of the external force, that is, when its state changes from outside to inside, or vice versa. We then precisely locate the point where it crosses the boundary via dichotomic subdivision as described below.

If the current resolution of deformation $(\varepsilon_{\min}/2)$ does not match the resolution of the silhouette images, it is even possible that an IN or OUT vertex may cross the boundary several times at one single iteration, jumping over and missing fine shape details. To prevent this, before moving an IN or OUT vertex P_0 with the external force, we sample its motion trajectory, which is of length $\varepsilon_{\min}/2$, at the maximum available resolution. If a zero-crossing is detected at any of these sampled points, say P'_0, the displacement of the vertex is refrained at that point and a dichotomic subdivision is carried out to search for the point P where the isolevel function $f(P)$ is zero on the line segment joining the point P'_0 and the initial position P_0. A sufficiently small threshold value ξ, $-\xi < f(P) < \xi$, is used to

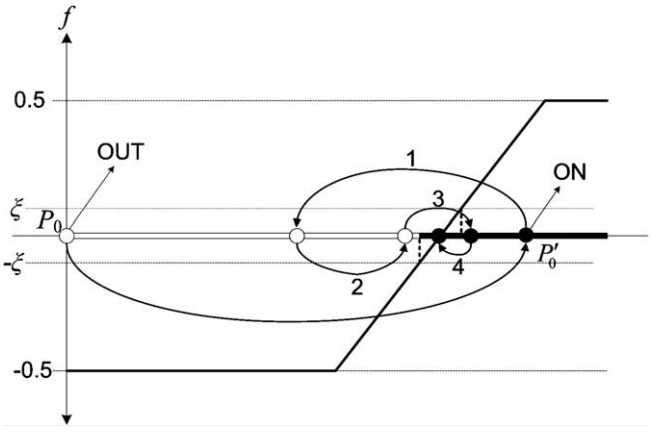

Fig. 1. Fine-tuning. For this example, the binary search takes four steps to locate the boundary point.

determine how close the isolevel of a given point must be to assume it as being exactly on the boundary surface (see Fig. 1). The vertex is then moved onto this location. Note that the vertex is not frozen at this point and can still move due to the internal forces at the next iterations. Hence the whole deformable mesh continues to deform under the influence of the external force and the fine tuning procedure until it finds its optimal placement within the narrow band around the object boundary, trying to also satisfy the smoothness constraint imposed by the internal force.

3.3. Adaptive resolution

The choice of the minimum edge length, $\varepsilon_{\min}$, is critical. Ideally it should match the resolution of the silhouette images, that is, the finest detail that can be captured. The smaller the value of $\varepsilon_{\min}$, the higher is the resolution, hence the more is the flexibility of the deformable mesh. But the choice of $\varepsilon_{\min}$ also scales and puts an upper bound to the magnitude of the external force. Thus small values of $\varepsilon_{\min}$ slow down the algorithm. To speed up the algorithm, one may choose to initially set $\varepsilon_{\min}$ to a relatively large value and then gradually increase the resolution by decreasing the value of $\varepsilon_{\min}$ so as to recover more and more fine details of the shape.

The refinement of the deformable mesh can be carried out in an adaptive manner. Once the deformation algorithm converges to the optimal surface for a given $\varepsilon_{\min}$, the current value of $\varepsilon_{\min}$ is reduced by a factor κ such that $\varepsilon_{\min} \leftarrow \kappa\varepsilon_{\min}$ and the surface evolution is reiterated. At this reiteration, only the edges of the mesh which still remain outside or inside of the boundary band are subjected to further restructuring operations with the new decreased value of $\varepsilon_{\min}$. To determine whether an edge remains outside or inside, we sample it at the desired finest resolution, e.g., the resolution that matches the resolution of the silhouette images, and then check whether any of the points sampled along the edge is OUT or IN. Hence the deformable mesh is refined only at parts where the current resolution does not suffice to represent the local shape details. Recalling that $\varepsilon_{\max} = 2\varepsilon_{\min}$, the factor κ to reduce the resolution is to be selected as $0.5 < \kappa < 1$ for a robust and smooth resolution transition. The resolution of the deformable mesh is gradually increased by this factor until no edge remains outside or inside the boundary band, or interactively until a satisfactory visual quality is achieved.

4. Topology control

The deformable model, which is initially zero genus, should be able to modify its topology whenever necessary during its deformation so as to recover objects with arbitrary non-zero genus. Our strategy to control topology involves two tasks. First, any self-intersection of the deformable mesh is to be avoided, which requires a collision detection mechanism. Second, the self-intersections which are due to a need for topology modification are to be identified by exploiting the explicit topology information available in the silhouette images.

4.1. Collision handling

We develop an efficient collision detection mechanism based on the minimum and maximum edge length constraints imposed on the deformable mesh. We first note that neighboring vertices never create self-intersections due to the minimum edge length constraint which also bounds the displacement of a vertex at one iteration by half of the minimum edge length, $\varepsilon_{\min}/2$. Hence the basic idea in collision detection is to prevent non-neighboring vertices from approaching each other by more than some distance threshold δ.

The procedure that we use to handle collisions is as follows. All the vertices of the deformable mesh are first displaced by applying the external force. Then each vertex is checked one by one against the non-neighboring vertices. If a vertex is found to have approached any other vertex by more than the collision detection threshold δ, then a collision is detected and the vertex is moved back to its original position. The value of the collision threshold δ depends on the maximum edge length parameter $\varepsilon_{\max}$ and the displacement bound $\varepsilon_{\min}/2$. To set this threshold, we consider the worst scenario where the position of a vertex is checked against the largest possible triangle on the deformable mesh, which is an equilateral triangle of side $\varepsilon_{\max} = 2\varepsilon_{\min}$, as visualized in Fig. 2. The vertex must not approach to any point inside this triangle by more than the maximum possible displacement $\varepsilon_{\min}/2$ since otherwise, at the next iteration, the triangle and the vertex may move at opposite directions, intersecting each other and falling apart by more than $\varepsilon_{\min}/2$ distance. Note that the centroid of the triangle is the farthest interior point from all three corners of the triangle with distance $2\varepsilon_{\min}/\sqrt{3}$ to each of the corners. The collision zone starts at the point whose vertical distance from the centroid is $\varepsilon_{\min}/2$. The distance of this point to each of the triangle vertices is given by the diagonal length of the right triangle formed by this point, the centroid and a triangle vertex. Hence the collision threshold δ must satisfy the following inequality,

$$\delta > \sqrt{\left(\frac{2\varepsilon_{\min}}{\sqrt{3}}\right)^2 + \left(\frac{\varepsilon_{\min}}{2}\right)^2} = \sqrt{\frac{19}{12}}\,\varepsilon_{\min}. \tag{7}$$

By using an octree structure where each vertex is associated with a node (or voxel) and by checking each vertex only against those in its vicinity, the collision detection algorithm can be implemented in an efficient manner with $O(n \log n)$ complexity, where n is the number of vertices in the deformable mesh. We further reduce the complexity of the algorithm by checking only the active OUT vertices for any possible collision. We note that, due to the adaptive resolution scheme and also since the maximum edge length constraint is a soft requirement, there may be active triangles on the deformable mesh with sides longer than $\varepsilon_{\max}$. We handle these large triangles by sampling, that is, by virtually quadrisecting each triangle in a recursive manner until the maximum edge length requirement is met. The "phantom" vertices which are virtually sampled on such large triangles are used only for collision detection purposes.

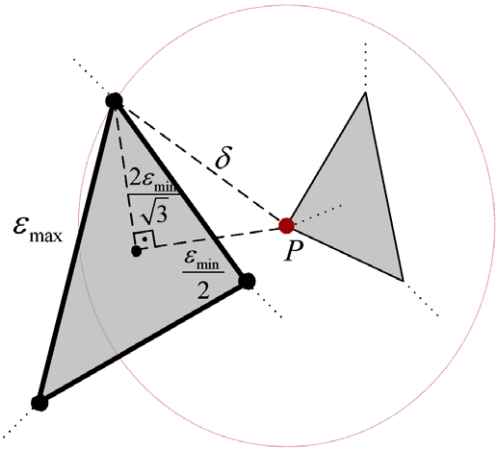

Fig. 2. Collision detection. The vertex P is checked at the worst case against the largest possible triangle on the mesh, that is, an equilateral triangle of side $\varepsilon_{\max}$. If the vertex approaches to any vertex of the mesh by more than the threshold δ, a collision occurs.

The collision handling method described above may have a negative impact on evolution of the deformable mesh if nearby but non-neighboring vertices are strictly refrained to approach each other by more than the specified collision threshold. To improve flexibility of the deformable mesh, the neighborhood in which vertices are allowed to freely move can be extended to cover a larger area. In our implementations, we have defined this area as the two-ring neighborhood of a vertex assuming that self-intersections are not likely to occur for very nearby vertices due to the regularizing effect of the internal force as well as the minimum edge length constraint.

4.2. Topology modification

Collisions (or self-intersections) between distant parts of a deformable mesh do not always indicate a need for topology change as demonstrated in Fig. 3, where we display two instances of surface evolution for two different synthetic objects. Although the first shape is zero genus, severe collisions occur between distant parts of the deformable model due to the elongated nature of the shape whereas the collisions for the second object with genus one are mostly due to a need for topology modification. We differentiate these two distinct cases by exploiting the topology information that can be inferred from silhouettes.

Object silhouettes often contain useful information about the topology. In Fig. 3, we observe that some silhouette images of the second object with genus one contain holes whereas the silhouettes of the other do not. We define a silhouette hole as a connected component that is bounded by the object silhouette in the image, that is, a region that belongs neither to the object nor to the background. Based on the hole information available in the silhouette images and the current status of the deformable mesh, we define a set of necessary conditions that a pair of vertices has to meet to be considered for topology modification (merging), which we will refer to as "the merging conditions":

1. The vertices must collide with each other.
2. Both vertices must be OUT.
3. The deformable mesh must have already converged at the current resolution.
4. Each of the vertices, when projected, must fall inside a hole in at least one of the silhouette images.
5. Each of the vertices, when projected, must not fall into the background region in any of the silhouette images, that is, into the region which is neither inside the object silhouette nor inside a hole.

When a pair of vertices that satisfies the above merging conditions is detected, the surface patches around them can be combined via topology merging. The colliding vertices are first removed from the mesh structure. Their neighboring vertices are then matched and connected so as to build a surface tunnel between two separate parts of the deformable surface (Lachaud and Taton, 2003). In order to facilitate the process of vertex matching, before the merging operation, we equalize the valences of the colliding vertices by applying collapse operation(s) at the vertex whichever has a valence higher than the other. The topology merging process is depicted in Fig. 4 on an example.

A topology merging operation creates new triangles that may possibly intersect with the existing geometry if no precaution is taken. This may happen particularly if more than two surface parts are in collision with each other at the same time and location. The most straightforward precaution to take for avoiding this type of self-intersections is to allow merging between two vertices only if they are not in collision with some other vertex that is not neighboring to any of the collided vertices (taking into account also the phantom vertices defined in Section 4.1). Note that, when two surfaces collide at convergence at a given resolution level, they usually collide at multiple locations and the merging operation can be performed at any of these locations. On rare occasions where two surfaces collide only at one single vertex pair which also happens to

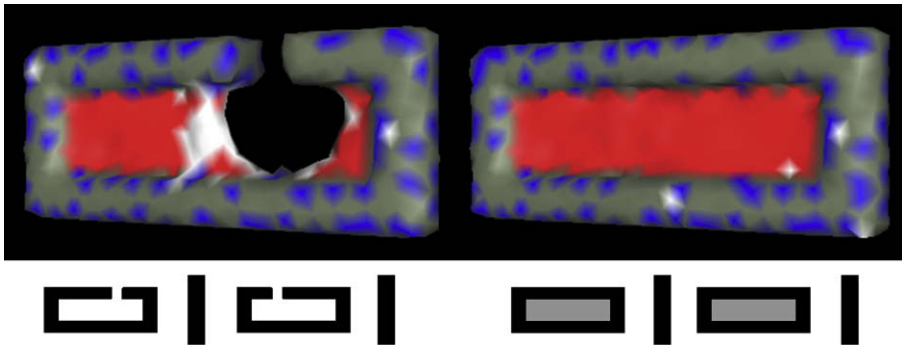

Fig. 3. Two instances of surface evolution for two different objects. The vertices in collision are marked in red. The synthetic silhouettes used for reconstruction of the two objects are displayed on the bottom where the holes are marked in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

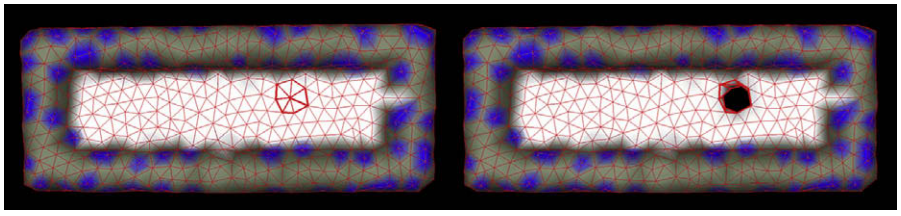

Fig. 4. Topology merging. A vertex pair satisfying the merging conditions is detected on the deformable mesh, and the two parts of the deformable surface are then merged at the colliding vertices.

be in collision with some other surface, the merging operation is delayed to the next level of resolution so that it can safely be performed on a finer triangle mesh.

4.3. Surface reconstruction algorithm

In Fig. 5, we provide the block diagram of the overall reconstruction algorithm. The deformable mesh, initially at a relatively coarse resolution, starts its evolution towards the object boundary, avoiding collisions but with topology merging disabled. The surface evolution algorithm is iterated until convergence, that is, until the deformable mesh finds its optimal positioning at the current resolution so that the vertices of the model no longer move. We employ an adaptive smoothing strategy during mesh deformation. We use the tangential Laplacian to smooth IN and OUT vertices whereas ON vertices are smoothed by the internal force defined in Section 2 as a combination of tangential Laplacian and surface fairing. This adaptive strategy improves the flexibility of the model during deformation, yet finally resulting in a smooth mesh model. In order to speed up convergence, the ON vertices which are detected to no longer move at any iteration of the deformation algorithm are deactivated.

At convergence a decision is to be made: Either the algorithm is terminated, or the resolution of the deformable mesh is adaptively increased, or a topology merging operation is applied. If there exists a pair of vertices that satisfies the merging conditions given in Section 4.2, we enable topology merging and modify the genus by applying the topology merging operator only once. We then disable topology merging and reiterate the surface evolution algorithm until convergence, i.e., the hole on the boundary surface is totally recovered. If there exist OUT and IN vertices at convergence, but no colliding vertices that satisfy the merging conditions, the value of $\varepsilon_{\min}$ is decreased by the factor κ (which we always set as 2/3) and the surface evolution algorithm is reiterated so as to adaptively refine the resolution and adjust the positioning of the deformable mesh as described in Section 3.3. These reiterations of the surface evolution algorithm are repeated, each time either by increasing the resolution or by recovering a hole on the surface. Note that since the topology, which is initially of genus zero, is recovered step by step in a controlled manner, we do not need any topology split operations. The algorithm terminates whenever the maximum available resolution, which is limited with the resolution of the silhouette images in our case, is achieved or the deformable model precisely fits to the surface data so that no IN or OUT edges remain on the final mesh. The overall algorithm is demonstrated on a synthetic test object with genus two in Fig. 6.

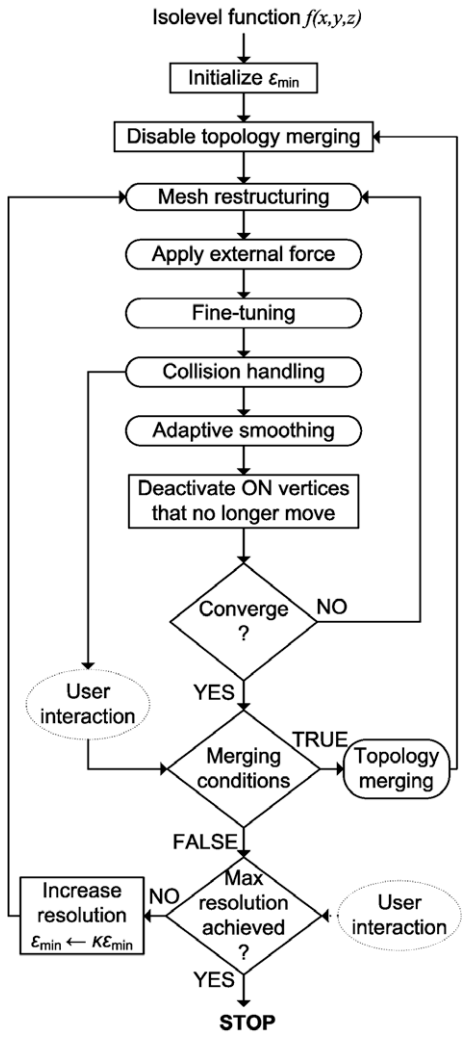

Fig. 5. The block diagram of the overall reconstruction algorithm.

To speed up the reconstruction algorithm as well as to increase robustness, an optional simple user interaction mechanism can be incorporated into the algorithm. At the convergence of each reiteration of the surface evolution algorithm, the deformable model can be displayed on the monitor, as in Fig. 6, so as to reveal the current labeling (OUT, IN or ON) of the vertices and to check whether there exists any vertex pair satisfying the merging conditions. The user can then decide to continue either by enabling topology merging or by adaptive mesh refinement. Such a user interaction mechanism is very effective especially when the object shape is very complicated in topology and exhibiting severe self surface occlusions so that the silhouette information does not enable exact recovery of the topology. An example for which the merging conditions given in Section 4.2 may not be sufficient and hence user input is necessary is when a hole itself contains very fine details that cannot be captured at the current resolution. In such a case, the hole can be recovered correctly only by increasing the resolution without enabling topology merging, though in our experiments we have never encountered such cases. We finally note that the user can also choose to provide input to terminate the algorithm at an earlier stage when a satisfactory visual quality is achieved.

4.4. Limitations

The merging conditions defined in Section 4.2 provide a heuristic and practical way of controlling topology using merging operations. Although our method can correctly recover the topology for most ordinary real-world objects based on these conditions, there exist some limitations which can be classified into two distinct cases. The first case of failure is due to the fact that some types of surfaces with special structure, such as knot surfaces, cannot be recovered from deformation of the bounding sphere by using only the merging operator. This failure case is demonstrated on an example in Fig. 7, where we consider a torus knot which can be obtained from a sphere by applying at least one topology splitting operation along with two merging operations. Hence, by incorporating a topology splitting operator into our topology control scheme, such as the one described in (Lachaud and Taton, 2003), we think that our method can be extended to handle surfaces with arbitrary topology without such restrictions.

The second case of failure may occur in some rare situations when the deformable mesh converges but we cannot find any vertex pair that satisfy the merging conditions, though a topology merging operation is necessary. This happens particularly if the

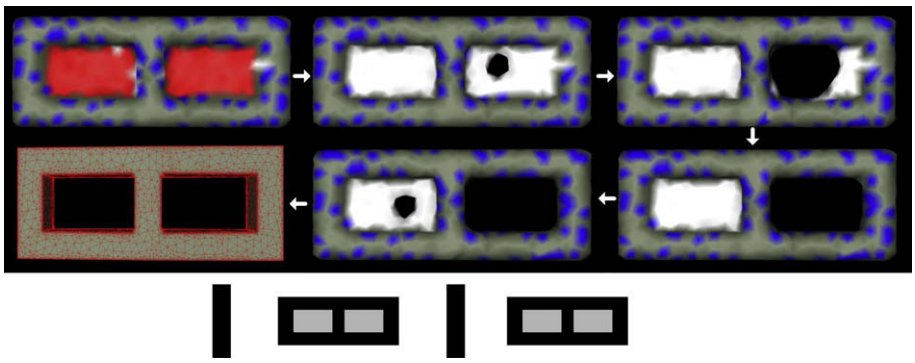

Fig. 6. Demonstration of the overall reconstruction algorithm on a synthetic object of genus two. The first image shows the coarse resolution deformable mesh at convergence, where the vertices satisfying the merging conditions are marked in red. The two holes on the surface are recovered by two successive reiterations of the deformation algorithm and two topology merging operations. The deformable mesh is then gradually refined in an adaptive manner until there remain no IN and OUT vertices on the final reconstructed mesh. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

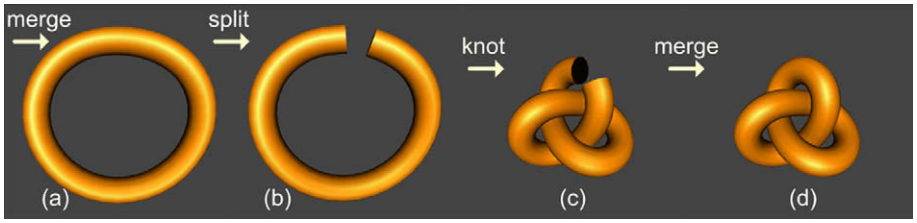

Fig. 7. Transforming a sphere into a torus knot with two topology merging and one splitting operations. (a) A torus is first obtained from the sphere by topology merging. (b–d) The torus should be split, tangled and then merged again to obtain a torus knot.

background regions in the silhouette images, when reprojected into 3D space, span the totality of a hole so that no point inside the hole ever meets the 5th merging condition. This situation can be illustrated with the following example. Consider a perfect torus which is symmetrically bent along one of its axes and viewed from three orthogonal directions aligned with the axes of the torus. In this case the silhouettes will be respectively a ring and two "C" shapes, and no vertex pair will ever meet the 5th condition. Fortunately, such situations are very unlikely to occur in practice with real surfaces which are never as perfect as purely geometrical objects, hence there usually remain points inside a hole, which are not covered by the background reprojections. As a matter of fact, we have not encountered this failure case in any of our experiments.

5. Experimental results

We have tested our deformation-based reconstruction technique on the silhouette images of three different objects. The original images of these objects, namely the Cup (made of stone), the Elephant (made of wood) and the synthetic Dragon (from Stanford 3D Scanning Repository (Curless and Levoy, 1996)) are displayed in Fig. 8. The resolution of images used for reconstructions are 2000×1310 for the Cup and Elephant objects, and 1024×768 for the synthetic Dragon. The image sequence for each object contains 72 equally-spaced horizontal views sampled from a complete

Fig. 8. Original images of the Elephant, Cup and the synthetic Dragon objects.

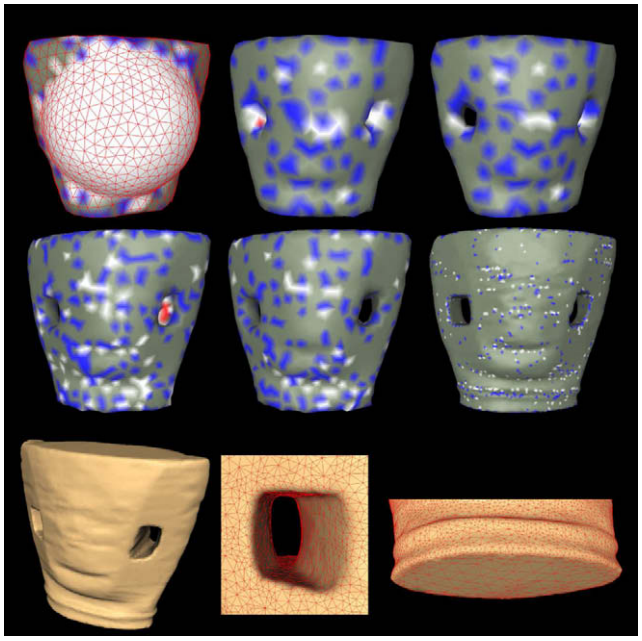

Fig. 9. Reconstruction of the Cup. (From left to right) (First row) The deformable model at various iterations of the deformation process at the initial resolution. At convergence there is only one single vertex pair that satisfies the merging conditions (marked in red). The corresponding hole is recovered at this resolution. (Second row) The other hole is detected and recovered at the second level of detail. The last image shows the surface at the highest (5th) level of detail. (Third row) Views from the final reconstructed mesh (approximately 29 K triangles). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

rotation. Our acquisition setup consists of a CCD camera, a turntable, a computer and a backdrop for silhouette extraction. The camera is calibrated and the silhouettes are extracted via the background saturation method described in (Yemez and Schmitt, 2004). In the synthetic case, the silhouettes of the Dragon have artificially been created using perspective projection, and hence the exact camera parameters are known a priori.

In Figs. 9–11, we illustrate the silhouette-based reconstructions for the three objects. For each reconstruction, we provide views of the deformable model at various iterations of the surface deformation process as it evolves starting from the bounding sphere towards the object boundary. The bounding sphere is automatically determined from silhouette images and represents the initial coarse deformable mesh with $\varepsilon_{\min} = 0.06$ for the Cup object and with $\varepsilon_{\min} = 0.04$ for the others. The value of $\varepsilon_{\min}$ is specified as the ratio of the minimum edge length to the radius of the bounding sphere for each object. Note that, since we usually terminate the reconstruction algorithm before the maximum available resolution is achieved, the final meshes, though they exhibit a very good visual quality, still contain some few IN and OUT vertices.

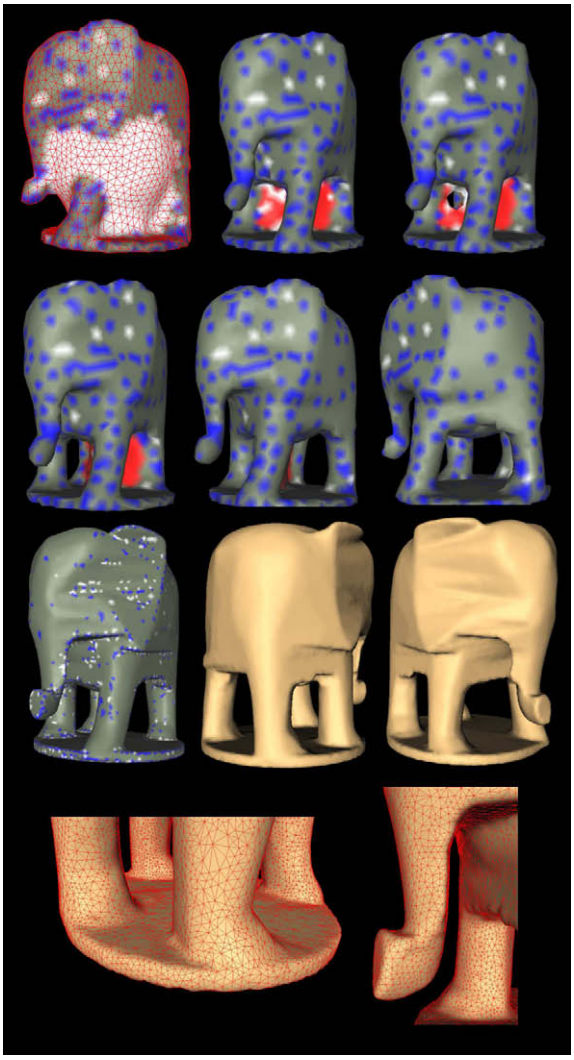

Fig. 10. Reconstruction of the Elephant. (From left to right, top to bottom) The deformable model at six various iterations of the deformation process at the initial resolution, and various views from the final reconstructed mesh (approximately 37 K triangles). Note that the three-genus topology is recovered at the initial resolution by three successive reiterations of the deformation algorithm and three topology merging operations.

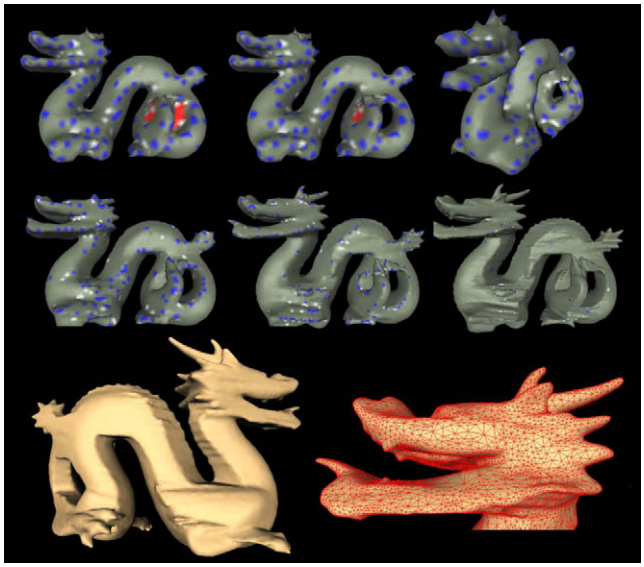

Fig. 11. Reconstruction of the Dragon. (From left to right) (First row) The deformable model at three various iterations of the deformation process at the initial resolution. (Second row) 2nd, 3rd and 5th levels of detail. (Third row) Various views from the final reconstructed mesh (approximately 43 K triangles).

The first object, the Cup, has a relatively simple shape. However it contains two deep holes on the surface and hence constitutes a good example of how the topology of a shape can be recovered in a robust manner using our method. Fig. 10 displays the reconstruction of the Elephant object which has a complicated structure of holes all connected to each other. The topology of the shape is of genus three, which can correctly be recovered with our method by applying the topology merging operator three times at the initial low resolution. In Fig. 11, we illustrate the reconstruction of the Dragon which has a very complex shape with holes and very fine surface details. Note especially how the thin tube-like structures on the head of the Dragon are gradually recovered as the resolution increases. The final model is smooth and contains only a very small number of IN and OUT vertices.

In Table 1, we provide the number of surface evolution iterations and the execution time needed for convergence at each level of detail for each object. The execution times are measured on a notebook computer with 2.2 GHz Intel T7500 processor. The increasing levels of detail are obtained by setting the parameter $\varepsilon_{\min}$ to 0.060, 0.040, 0.027, 0.018, 0.012 and 0.008, respectively. The choice of the coarsest and the highest levels of detail varies from one object to another depending on the shape complexity.

We compare our deformation-based reconstruction method with the volumetric visual hull technique described in (Yemez and Schmitt, 2004), which makes use of an intermediate octree representation followed by marching cubes triangulation. This

Table 1
Number of iterations and execution time at each (increasing) level of detail of the reconstructions for each object.

Model	Iterations (#)					
Cup	26	13	8	6	5	–
Elephant	–	78	6	6	5	–
Dragon	–	87	13	13	7	8
	Time (s)					
Cup	32	24	42	69	118	–
Elephant	–	207	27	79	134	–
Dragon	–	293	87	116	131	229

technique produces very accurate and topologically consistent mesh models when coupled with the marching cubes implementation described in (Lewiner et al., 2003), however it suffers from topological ambiguities especially under noise. In Fig. 12, we display the reconstructions of the synthetic Dragon obtained with this technique and our method under noisy conditions. We consider two different resolution levels for the volumetric method, where the depth of the octree structure is $R = 8$ and $R = 9$, respectively. To simulate the noise of the silhouette extraction process, we corrupt the vertices of the original Dragon model with random noise before creating its silhouette projections. We consider two cases where the amplitudes of the noise vector are 2% and 3% of the radius of the bounding sphere. We observe that in all cases, our method produces significantly smoother and topologically correct mesh representations whereas the other method results in reconstructions with surface artifacts, geometric distortions and disconnected surface parts even in the noiseless case (see also Fig. 13). In Fig. 14, we compare the reconstructions obtained by the two methods at some lower resolution, i.e., with octree depth $R = 5$. We observe that volumetric sampling of the surface cannot resolve the topology at this resolution and although the marching cubes algorithm produces a consistent triangulation, the resulting mesh model is not topologically correct whereas in our case we correctly recover the topology at a comparable resolution.

In the case of the synthetic Dragon, since the original mesh models are available, one can measure and compare the accuracy performances of the two methods. We quantify the reconstruction error as the average distance of the vertices of the original model (very high in resolution) to the surface of the reconstructed model. The obtained reconstruction errors, when the models are normalized into unit sphere, comes out to be as given in Table 2. We observe that our method yields smoother reconstructions with almost the same accuracy (the volumetric method is only slightly better). As for the computational complexity, our reconstruction method takes 856 sec whereas the volumetric method takes 541 and 2440 seconds, respectively for $R = 8$ and $R = 9$. We note that the volumetric technique described in (Yemez and Schmitt, 2004) does not employ an adaptive sampling strategy such as the one in (Erol et al., 2005), which could achieve high level of detail rep-

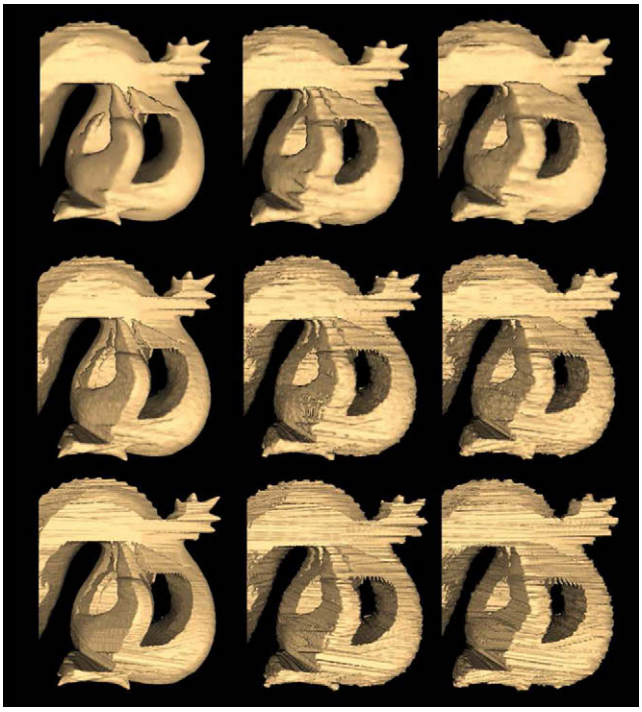

Fig. 12. Performance comparison under noise. Reconstruction of the Dragon using our method (first row) and the method in (Yemez and Schmitt, 2004) at two resolutions $R = 8$ (second row) and $R = 9$ (third row) with increasing amplitudes of random noise (from left to right, noiseless, 2% and 3%, respectively).

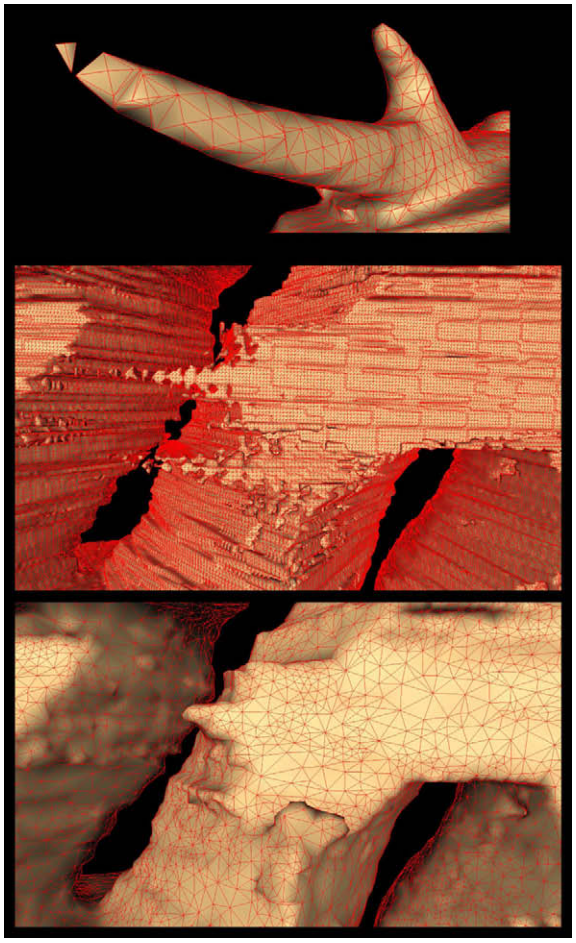

Fig. 13. Zoom on the mesh models reconstructed (first row) with the volumetric method at $R = 7$ in the noiseless case, (second and third rows) with the volumetric method at $R = 9$ and our method under 3% random noise, respectively.

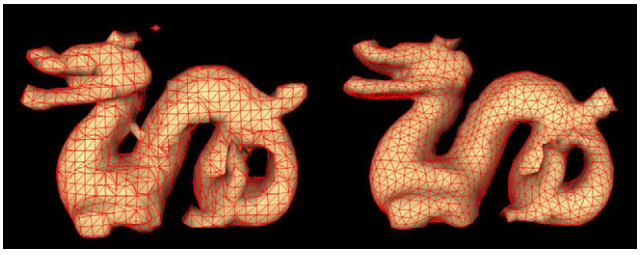

Fig. 14. Reconstruction of the Dragon at low resolution using the volumetric method (left) and our method (right).

Table 2
Reconstruction errors obtained with the volumetric method at two different resolutions and with our method at varying noise levels.

	Noiseless	Noisy (2%)	Noisy (3%)
Mesh deformation	0.011	0.017	0.023
Volumetric ($R = 8$)	0.010	0.016	0.021
Volumetric ($R = 9$)	0.010	0.015	0.020

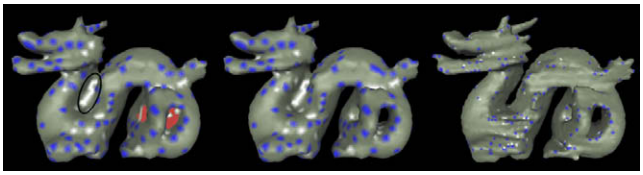

Fig. 15. Topology control on the reconstruction of the Dragon under 3% random noise. The holes are correctly detected and recovered at the initial resolution whereas the collided region (marked with an ellipse) is recovered by adaptively increasing the resolution two times with merging disabled.

resentations in better execution times than the ones presented here. Nevertheless, among various other volumetric techniques existing in the literature for the shape from silhouette problem, the one that we picked in this work for comparison with our method provides a good demonstration of the typical problems encountered with the volumetric approach. Although some volumetric techniques may perform better than the others in terms of efficiency, accuracy or correctness, the problems of smoothness and topological ambiguity remain inherent to all these techniques.

In Fig. 15 finally, we demonstrate the effectiveness of our topology control mechanism on the noisy Dragon reconstruction. At the initial coarse resolution, the deformable mesh cannot penetrate into the region marked with an ellipse on the figure due to a very narrow gap. As a result, severe collisions, which are not due to a need for topology change, occur at convergence on this part of the surface. However, based on the silhouette information, these collisions are successfully differentiated from those for which topology merging is actually required, and the overall topology is then correctly recovered by adaptively increasing the resolution.

6. Conclusion

We have presented a computationally efficient shape from silhouette method based on mesh deformation, which can produce accurate 3D mesh models of complex real objects with arbitrary genus. The method performs well and yields smooth reconstructions also in the presence of noise. By exploiting the explicit topology information available in the silhouette images, topology modifications can be applied to the deformable surface whenever necessary in a robust and controlled manner. We think that by appropriately defining the external forces, our topology control strategy can also be applied to other types of surface data provided that an isolevel function is available, such as biomedical data where the slices of a 3D image also contain useful topology information in a similar way to silhouettes. In the cases where the explicit topology information is not sufficient, as might sometimes be in the case of silhouettes, or difficult to extract, a simple user interaction mechanism can be incorporated into the topology control scheme.

There are advantages and disadvantages of the proposed shape from silhouette technique over volumetric methods. First, it yields smoother surface representations especially in the presence of noise. Second, it can correctly recover the topology for most of the typical real-world objects with arbitrary genus even at low resolutions while volumetric methods always have to deal with topological ambiguities. As briefly discussed in Section 4.4, the main disadvantage of our method is that it fails to reconstruct some special types of surfaces such as knot surfaces while volumetric visual hull techniques do not usually have such restrictions. Also, volumetric techniques can be computationally more efficient, though this efficiency difference is not drastic.

Another advantage of the proposed method is that it can be applied to the problem of shape from silhouette across time for dynamic scene modeling, as we plan to address as future work. Time-varying mesh representations with connectivity as fixed as possible, but with changing vertex positions, would certainly provide enormous efficiency for storage, processing and visualization. There have been very few attempts to achieve such time-consistent representations such as in (Mueller et al., 2004; Bilir and Yemez, 2008), but these works are yet quite premature and can obtain time-consistent meshes only for very short time intervals. Since our method is based on mesh deformation, the connectivity information can be tracked through iterations and hence the presented method can be employed for building efficient time varying surface representations.

References

Bilir, S.C., Yemez, Y., 2008. Time varying surface reconstruction from multiview video. In: IEEE Internat. Conf. on Shape Modeling and Applications (SMI'08), pp. 47–51.

Boyer, E., Franco, J.-S., 2003. A hybrid approach for computing visual hulls of complex objects. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 695–701.

Carranza, J., Theobalt, C., Magnor, M.A., Seidel, H., 2003. Free-viewpoint video of human actors. ACM Trans. Graph. 22 (3), 569–577.

Chien, C.H., Aggarwal, J.K., 1986. Identification of 3D objects from multiple silhouettes using quadtrees/octrees. Comput. Vision Graphics Image Process. 36 (2–3), 256–273.

Curless, B., Levoy, M., 1996. A volumetric method for building complex models from range images. In: Proc. SIGGRAPH'96, pp. 303–312.

Duan, Y., L.Yang, Qin, H., Samaras, D., 2004. Shape reconstruction from 3D and 2D data using pde-based deformable surfaces. In: Proc. ECCV, pp. 238–251.

Erol, A., Bebis, G., Boyle, R.D., Nicolescu, M., 2005. Visual hull construction using adaptive sampling. IEEE Workshops Appl. Comput. Vision, 234–241.

Esteban, C.H., Schmitt, F., 2004. Silhouette and stereo fusion for 3D object modeling. Computer Vision and Image Understanding 96 (3), 367–392.

Fang, Y.-H., Chou, H.-L., Chen, Z., 2003. 3D shape recovery of complex objects from multiple silhouette images. Pattern Recognition Lett. 24 (9–10), 1279–1293.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1993. Mesh optimization. In: Proc. SIGGRAPH'93, pp. 19–26.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. Internat. J. Comput. Vision 1 (4), 321–332.

Kobbelt, L.P., Bareuther, T., Seidel, H., 2000. Multiresolution shape deformations for meshes with dynamic vertex connectivity. Comput. Graphics Forum (Eurographics'00) 19, C249–C260.

Lachaud, J.-O., Taton, B., 2003. Deformable model with adaptive mesh and automated topology changes. In: Internat. Conf. on 3D Digital Imaging and Modeling (3DIM'03), pp. 12–19.

Lewiner, T., Lopes, H., Vieira, A., Tavares, G., 2003. Efficient implementation of marching cubes' cases with topological guarantees. J. Graphics Tools 8 (2), 1–15.

Mueller, K., Smolic, A., Merkle, P., Kautzner, M., Wiegand, T., 2004. Coding of 3D meshes and video textures for 3D video objects. In: Proc. Picture Coding Symposium.

Newman, T.S., Yi, H., 2006. A survey of the marching cubes algorithm. Comput. Graphics 30 (5), 854–879.

Niem, W., Wingbermuhle, J., 1999. Automatic reconstruction of 3D objects using a mobile camera. Image Vision Comput. 17 (2), 125–134.

Park, J.-Y., McInerney, T., Terzopoulos, D., Kim, M.-H., 2001. A non-self-intersecting adaptive deformable surface for complex boundary extraction from volumetric images. Comput. Graphics 25, 421–440.

Szeliski, R., 1993. Rapid octree construction from range sequences. Comput. Vision Graphics Image Process. 58 (1), 23–32.

Tarini, M., Callieri, M., Montani, C., Rocchini, C., 2002. Marching intersections: An efficient approach to shape-from-silhouette. Vision Modeling and Visualisation VMV'2002, 283–290.

Taubin, G., 1995. A signal processing approach to fair surface design. In: Proc. SIGGRAPH'95, pp. 315–358.

Terzopoulos, D., Witkin, A., Kass, M., 1991. On active contour models and balloons. CVGIP: Image Understanding 53, 211–218.

Wood, Z.J., Schröder, P., Breen, D., Desbrun, M., 2000. Semi-regular mesh extraction from volumes. In: Proc. Visualization'00, pp. 275–282.

Yemez, Y., Schmitt, F., 2004. 3D reconstruction of real objects with high resolution shape and texture. Image Vision Comput. 22, 1137–1153.

Yemez, Y., Wetherilt, C.J., 2007. A volumetric fusion technique for surface reconstruction from silhouettes and range data. Computer Vision and Image Understanding 105 (1), 30–41.