Shape Reconstruction from 3D and 2D Data Using PDE-Based Deformable Surfaces

Ye Duan[1], Liu Yang[2], Hong Qin[2], and Dimitris Samaras[2]

[1] Department of Computer Science
University of Missouri at Columbia, Columbia, MO 65203, USA
duanye@missouri.edu,
http://www.cs.missouri.edu/~duan

[2] Center for Visual Computing, Department of Computer Science
State University of New York at Stony Brook, NY 11794, USA
{yliu, qin, samaras}@cs.sunysb.edu
http://www.cs.sunysb.edu/~{yliu,qin,samaras}

Abstract. In this paper, we propose a new PDE-based methodology for deformable surfaces that is capable of automatically evolving its shape to capture the geometric boundary of the data and simultaneously discover its underlying topological structure. Our model can handle multiple types of data (such as volumetric data, 3D point clouds and 2D image data), using a common mathematical framework. The deformation behavior of the model is governed by partial differential equations (e.g. the weighted minimal surface flow). Unlike the level-set approach, our model always has an explicit representation of geometry and topology. The regularity of the model and the stability of the numerical integration process are ensured by a powerful Laplacian tangential smoothing operator. By allowing local adaptive refinement of the mesh, the model can accurately represent sharp features. We have applied our model for shape reconstruction from volumetric data, unorganized 3D point clouds and multiple view images. The versatility and robustness of our model allow its application to the challenging problem of multiple view reconstruction. Our approach is unique in its combination of simultaneous use of a high number of arbitrary camera views with an explicit mesh that is intuitive and easy-to-interact-with. Our model-based approach automatically selects the best views for reconstruction, allows for visibility checking and progressive refinement of the model as more images become available. The results of our extensive experiments on synthetic and real data demonstrate robustness, high reconstruction accuracy and visual quality.

1 Introduction

During the past decade, PDE-driven surface evolution has become very popular in the computer vision community for shape recovery and object detection. Most of the existing work is based on the Eulerian approach, i.e., the geometry and topology of the shape is implicitly defined as the level-set solution of time-varying

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3023, pp. 238–251, 2004.

implicit functions over the entire 3D space [24], which can be computationally very expensive. In this paper, we propose a new PDE-based deformable model that, in contrast, takes the Lagrangian approach, i.e., the geometry and topology of the deformable surface are always explicitly represented throughout the simulation process. The elegance of our approach lies in the fact that we can use the same PDE-based model for different types of data. The only thing that is data-dependant is the control function, which describes the interaction with the data. This is an important property that will allow easy application of our methodology to other types of data, such as points, surfels, images and to incorporate other visual cues such as shading and optical flow.

Starting with [17], deformable models have achieved great success in the areas of computer vision and pattern recognition. In general, deformable models can be divided into two categories: explicit models and implicit models. Explicit models include parametric representations [27] and discrete representations [28]. Implicit models [24,5,14] handle topology changes, based on the modeling of propagating fronts, which are the level set of some scalar function. The desirable shape must be explicitly evaluated using marching-cube-like techniques [23] in an additional post-processing stage. The narrow band algorithm [14] can reduce the computational cost related to the higher dimension. Recently, topologically adaptive explicit models have been proposed [26,22], reviewed in detail in [25]. The aforementioned deformable models were proposed mainly for the purpose of shape reconstruction from volumetric data and for medical image segmentation. For shape reconstruction from point clouds, existing work is mostly on static methods. They are either explicit methods [11,1,8], implicit methods [15] or based on radial basis functions [4,10].

Compared with level-set based methods, our new model is simpler, more intuitive, and makes it easier to incorporate user-control during the deformation process. To ensure the regularity of the model and the stability of the numerical integration process, powerful Laplacian tangential smoothing, along with commonly used mesh optimization techniques, is employed throughout the geometric deformation and topological variation process. The new model can either grow from the inside or shrink from the outside, and it can automatically split to multiple objects whenever necessary during the deformation process. More importantly, our model supports level-of-details control through global subdivision and local/adaptive subdivision. Based on our experiments, the new model can generate a good, high-quality polygonal mesh that can capture underlying topological structure simultaneously from various datasets such as volumetric data, 3D unorganized point clouds and multiple view images. The explicit representation of the model enables us to check for visibility and camera pose directly.

Automatic reconstruction of 3D objects and environments from photographic images is important for many applications that integrate virtual and real data. [29] Many approaches have been used to solve the problem, e.g. by matching features [34] or textures [12]. In traditional stereo methods, many partial models must be computed with respect to a set of base viewpoints, and the surface patches must be fused into a single consistent model [2] by Iterative Closest Points(ICP) [30], but a parameterized model is still needed for final dense surface reconstruction, and there is no explicit handling of occlusion. Meshes and/or systems of particles [13] can be deformed according to constraints derived from images, but may end up clustering in areas of high curvature, and often fail with complicated topology. Recently, voxel-based approaches have been widely used to represent 3D shape [12,32,21,3,7] based on 3D scene space instead of image space. Marching-cube-like techniques [23] are still necessary to get the parameterized surface. The space carving method [21] recovers a family of increasingly tighter supersets of the true scene. Once a voxel is carved away, it cannot be recovered, and any errors propagate into further erroneous carvings. This is partially alleviated by probabilistic space carving [8]. The level set method [12,9,16] is based upon variational analysis of the objects in the scene and their images while handling topology changes automatically. To overcome the complexity of implicit representation of objects in the level set approach, [18,19] operate on a surface represented by a depth function, at the cost of being limited to surface patches. A multi-resolution method using space carving and level sets methods [33], starts with coarse settings, refined when necessary.

However these methods do not take advantage of the existence of a simple explicit geometric representation. We show that high quality results can be achieved with the use of a simple intuitive mesh, easy to interact with and can both incorporate all the available image information and also allow for progressive refinement as more images become available. With increased computer performance, our reconstruction method will soon achieve interactive run times. One can envision user controlling the quality of the reconstruction during image capture, being able to capture the most necessary remaining views to complete the reconstruction [31]. Our PDE-based deformable model is described in sec. 2 and applied to volumetric data in sec. 3.1, unorganized point clouds in sec. 3.2 and multi-view images in sec. 3.3. Experimental results on synthetic and real data are presented in sec. 4.

2 PDE-Based Deformable Surface

The deformation behavior of our new deformable surface is governed by an evolutionary system of nonlinear initial-value partial differential equations (PDE) with the following general form:

$$\frac{\partial S(p)}{\partial t} = F(t, k, k', f \cdots)U(p, t) \tag{1}$$

where F is speed function, t is the time parameter, k and k' are the surface curvature and its derivative at the point p, and f is the external force. $S(p, 0) = S_0(p)$ is the initial surface. U is the unit direction vector and often it represents the surface normal vector. Eq. 1 can be either directly provided by the user, or more generally, obtained as a gradient descent flow by the Euler-Lagrange equation of some underlying energy functionals based on the calculus of variations.

2.1 Model Refinement

Once an initial shape of the object is recovered, the model can be further refined several times to improve the fitting accuracy. In this paper, we have implemented two kinds of model refinement: global refinement and local/adaptive refinement. Global refinement is conducted by Loop's subdivision scheme [6].

Adaptive refinement is guided by fitting accuracy as measured by the variance of the distance from the triangle to the boundary of the object [36]. If the variance of the distance samples for a given triangle is bigger than a user defined threshold, then this triangle will be refined. The variance of a discrete set of distances is computed in the standard way: $V_T[d] = E[d^2] - E[d]^2$, where E denotes the mean of its argument. To calculate the variance of the distance samples for a given triangle, we temporarily quadrisect the triangle T into four smaller triangles and for each smaller triangle, calculate the distance at its barycentric center. At each level of adaptive refinement, all the triangles with fitting accuracy below the user-specified threshold will be quadrisected. The deformation of the model will resume only among those newly refined regions. In Fig. 4, we show different levels of refinements.

2.2 Mesh Regularity

To ensure that the numerical simulation of the deformation process proceeds smoothly, we must maintain mesh regularity so that the mesh's nodes have a good distribution, a proper node density, and a good aspect ratio of the triangles. This is achieved by the incorporation of a tangential Laplacian operator, and three mesh operations: edge split, edge collapse, and edge swap.

The tangential Laplacian operator is used to maintain good node distribution. The Laplacian operator, in its simplest form, moves repeatedly each mesh vertex by a displacement equal to a positive scale factor times the average of the neighboring vertices.

When edge lengths fall outside a predefined range, edge splitting and edge collapsing are used to keep an appropriate node density. Edge swapping is used to ensure a good aspect ratio of the triangles. This can be achieved by forcing the average valence to be as close to 6 as possible [20]. An edge is swapped if and only if the quantity $\sum_{p \in A} (valence(p) - 6)^2$ is minimized after the swapping.

2.3 Topology Modification

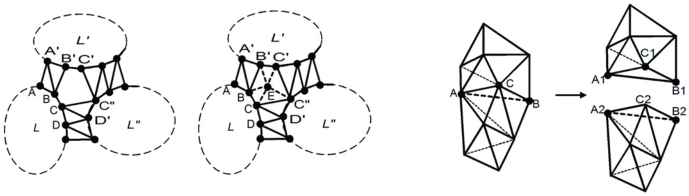
In order to recover a shape of arbitrary, unknown topology, the model must be able to modify its topology properly whenever necessary. In general, there are two kinds of topology operations: (1) Topology Merging, and (2) Topology Splitting.

Topology Merging. We propose a novel method called "lazy merging" to handle topology merging. The basic idea is that whenever two non-neighboring vertices are too close to each other, they will be deactivated. Topology merging will happen only after the deformation of the model stops and all the vertices become non-active. There are three steps in the topology merging operation: (1) *Collision Detection*, (2) *Merging-vertices Clustering*, and (3) *Multiple-contours Stitching*. **Collision Detection:** Collision detection is done hierarchically in two different levels: coarser-level and finer-level. Coarser-level collision detection is mainly for the purpose of collision exclusion. For each active vertex V, we will calculate its distance to all other non-neighboring active vertices. Vertices whose distance to the current vertex V is small enough so that no collision might happen between, will be passed to the finer-level collision detection. For each face f with three corner points (u, v, w) that is adjacent to one of the vertices being passed into the finer level of collision detection, we will calculate the distance between a number of sample points $\alpha u + \beta v + \gamma w$ of the face f with barycentric coordinates $\alpha + \beta + \gamma = 1$ and the current vertex V. If at least one of these distances is smaller than the collision threshold, the two corresponding vertices will be marked as merging vertices and will be deactivated. **Merging-Vertices Clustering:** After all the merging vertices have been deactivated, we need to divide them into several connected clusters. We randomly pick any merging vertex and find all of its connected merging vertices by a breadth-first search. We continue recursively, until all the merging vertices belong to appropriate merging vertex clusters. Then for each cluster of merging vertices, all the interior vertices will be removed and the remaining vertices will be put into a linked list. This is based on the following observation: *when two regions are merging together, only the boundary regions will remain, all the interior regions will be burned out (i.e. removed).*

Multiple-Contours Stitching: After the merging vertex linked lists have been created, we stitch them together in three separate steps:

1. For each vertex in the linked lists, find its closest vertex in other linked lists.
2. Based on the proximity information obtained from the previous step, find a pair of vertices A and B such that they are adjacent to each other in the linked list L (Fig. 1 (a)), and their closest merging vertices A' and B' are also adjacent to each other in the corresponding linked list L', in addition, the closest merging vertices of A' and B' are A and B, respectively. Starting from this pair of vertices A and B, iteratively go through the linked lists and if possible, connect each pair of adjacent vertices in one linked list to a corresponding vertex in another linked list and create a new triangle.
3. If there are more than two linked lists to be stitched together, then after stitching all the corresponding vertices, there may be some in-between gaps that need to be filled in. For example, in (Fig. 1 (a)), there is a gap between the linked lists L, L' and L" that consists of vertices B, C, C", C' and B'. We filled in the gap by creating a new vertex E at the center and connecting the new vertex E with all the other vertices in the loop (Fig. 1 (b)).

Topology Splitting. Topology splitting occurs when a part of the surface tends to shrink to a single point. In this scenario, the surface has to split up into two parts precisely at that location. We use a method similar to [22]. In particular, a split-operation is triggered if there exists three neighboring vertices which are



(a) (b) (c)

Fig. 1. (a)(b) Multiple contours stitching. (a) New triangles are created between corresponding vertices, and a gap is created by vertices $B, C, C", C', B'$. (b) The gap is filled in by creating a new vertex E in the center and connecting it with all the other vertices in the gap. (c) Topology split by splitting the virtual face ABC into two faces whose orientations are opposite to each other.

interconnected to each other, but the face formed by these three vertices does not belong to the model (i.e., a virtual face), and if the length of any of the three edges of the virtual face is smaller than the minimum edge length threshold and thus needs to be collapsed. For example, in Fig. 1(c), face ABC represents a virtual face that needs to be split. We divide the surface exactly at this location by cutting it into two open sub-surfaces. Then we close the two split-in-two surfaces using two faces $A_1B_1C_1$ and $A_2C_2B_2$ whose orientations are opposite to each other. Finally, we reorganize the neighborhood.

3 Surface Reconstruction

We have applied our PDE-based deformable surface to shape reconstruction from volumetric data, unorganized 3D point clouds and multi-view 2D images. The PDE we used is the general weighted minimal surface flow [5]:

$$\frac{\partial S}{\partial t} = (g(v + H) - \nabla g \cdot \boldsymbol{N})\boldsymbol{N}, \quad S(0) = S_0 \tag{2}$$

where $S = S(t)$ is the 3D deformable surface, t is the time parameter, and S_0 is the initial shape of the surface. Note that, H is the mean curvature of the surface, and N is the unit normal of the surface. v is a constant velocity that will enable the convex initial shape to capture non-convex, arbitrary complicated shapes. It is also useful to avoid allowing the model to get stuck into local minima during the evolution process. g is a monotonic non-increasing, non-negative function that is used for interaction of the model with the datasets, and will stop the deformation of the model when it reaches the boundary of the object. In essence, Eq. 2 controls how each point in the deformable surface should move in order to minimize the weighted surface area. Hence, the detected object is given by the steady-state solution of the equation: $S_t = 0$, i.e. when the velocity F is zero. In order to apply the model to different types of data, we simply need to provide the definition of g that is appropriate for the dataset. For example, in 2D multi-view based reconstruction, direct checking for visibility and camera pose can be easily incorporated into the appropriate control function.

3.1 Volumetric Images

For volumetric data sets, the stopping function g is defined as:

$$g(S) = \frac{1}{1 + |\nabla(G_\sigma * I(S))|^2} \tag{3}$$

where I is the volumetric density function, and $G_\sigma * I$ is the smoothed density function by convoluting with a Gaussian filter with variance σ.

3.2 Point Clouds

For surface reconstruction from 3D unorganized point clouds, we use a simplified version of Eq. 2 with part $\nabla g \cdot \boldsymbol{N}$ removed. The stopping function g is:

$$g(p) = \begin{cases} 1, & \text{if } D(p) < T_D \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $D(p)$ is the distance between current position p and its closest data points, T_D is the threshold distance that is decided by the sampling rate of the point clouds datasets. In order to efficiently find the closest data points of any given position p, we preprocess the point clouds by putting them into a uniform regular grid and connecting all the points inside one grid cell by a linked list. The above distance threshold T_D will stop the movement of the model before it arrives at the "real" boundary of the object. To reduce the distance from the model to the boundary of the object, after the model stops its deformation, we will project each vertex point to the local tangent plane of its k-nearest neighbors.

The local tangent plane can be estimated using principle component analysis (PCA) [15]: For any point p, its local tangent plane can be represented by a center point c and a unit normal vector n. The center point c is the centroid of the one neighborhood of point p, which is denoted as $Nbhd(p)$. The normal vector n is computed by doing eigenanalysis of the covariance matrix C of $Nbhd(p)$, which is a symmetric 3 x 3 positive semi-definite matrix:

$$C = \sum_{p_i \in Nbhd(p)} (p_i - c) \otimes (p_i - c) \tag{5}$$

Here, $\otimes$ denotes the outer product vector operator, and the normal vector n is the eigenvector associated with the smallest eigenvalue of the covariance matrix C. In our experiments, k is set to five.

3.3 2D Multiple Views

The new model is capable of recovering shape not only from 3D data but also from 2D images. In the case of 2D photographic images, we use a photo consistency as the weight function g in Eq. 2. Here we use photo consistency criterion similar to the one in [21]. We only calculate the photo consistency w.r.t. each of the model vertices. For numerical robustness, the consistency calculation is performed by projecting a small patch around each vertex, to the image planes. It is assumed that the patch is reasonably small w.r.t. the distance between the object and the camera, while it is large enough to capture local features over images. A reasonable planar approximation for Lambertian surfaces is to take a patch on the tangent plane to the vertex. Again, we can use PCA to estimate the normal of the tangent plane, which is actually the first order approximation for the current model surface. In order to find the correct correspondence of a point i_1 in correlation window A to a point i_2 in correlation window B (A and B are projections of surface patch P onto different images), we back-project i_1 to a point $p_1 \in P$ and then reproject p_1 to i_2 in correlation window B. Then we can calculate the photo consistency within patch projections in different views as discussed in [21]:

$$\begin{aligned} g = \sigma^2 = \sigma_R^2 + \sigma_G^2 + \sigma_B^2, \quad \sigma_R^2 = \tfrac{1}{N-1}\sum_i^N R_i^2 - (\tfrac{1}{N}\sum_i^N R_i)^2 \\ \sigma_G^2 = \tfrac{1}{N-1}\sum_i^N G_i^2 - (\tfrac{1}{N}\sum_i^N G_i)^2, \quad \sigma_B^2 = \tfrac{1}{N-1}\sum_i^N B_i^2 - (\tfrac{1}{N}\sum_i^N B_i)^2 \end{aligned} \tag{6}$$

where N is the number of selected views. We only select the best-N views for our reconstruction, ignoring the most degenerate views. For a particular camera, using the OpenGL depth buffer, we first check visibility based on the current model. The visibility info can be calculated for every iteration with the complexity O(mn), where m is the number of total cameras, and n is the number of surface vertices.

Among the visible points, we take those with the largest projection area to the image plane. The projection area gives both distance and pose information.

$$\frac{\delta O}{\delta I} = \frac{\cos\alpha}{\cos\theta}\left(\frac{Z}{f}\right)^2 \tag{7}$$

Using the geometric properties of solid angles [35], we derive our camera weight $w_{i,j}$ from Eq. 7

$$w_{i,j} = \tfrac{\cos\theta}{(\cos\alpha)^3}\left(\tfrac{f}{D}\right)^2, \quad w_{i,j} = 0 \text{ when not visible} \tag{8}$$

In Fig. 2, we show how to evaluate camera position and orientation.

Using Eq. 2, we can set the speed function for each vertex. We can start from a coarse mesh (like a cube), and subdivide it after it is close to the surface

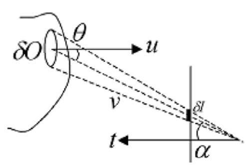


Fig. 2. Selecting the best-N views by camera positions

of the object. More specifically, we can use adaptive mesh refinement to locally subdivide areas with large curvature, where most details exist. The regularization term helps to keep the surface smooth and to make it converge to the final result. Since we have the explicit representation of the object surface, we can always incorporate new images taken from new camera positions, and incrementally enhance the quality of the final reconstruction. The complete algorithm is:

1. Preprocessing: For each camera position check visibility using the depth buffer value of each vertex.
2. For each vertex point on the model:
   a) For each camera, check camera visibility and pose.
   b) Select the best N-views.
   c) Calculate the photo consistency g at the vertex position (first order patch approximation).
   d) Calculate the gradient $\nabla(g)$ at the vertex position.
   e) Get the vertex speed from the PDE of Eq. 2, and the new vertex position in the next iteration.
3. Laplacian regularization.
4. Mesh adaptive refinement if necessary.

4 Experimental Results

Results from 3D Data: In this section, we will show experimental results on both real and synthetic datasets. In all the following figures, grey regions represent parts of the model that are still active and deforming, black regions represent deactivated parts of the model that have already reached the boundary of the object. In order to illustrate the good mesh quality generated by our new model, we will show at least one wireframe view of the model in all the following figures. The input volumetric dataset of Fig. 3 is obtained from CT scanning of a phantom of the vertebra. The data size is $128 \times 120 \times 52$ voxels. Fig. 4 and Fig. 5 illustrate the surface reconstruction process from 3D unorganized point clouds. The input dataset of Fig. 4 is the mannequin head with 7440 data points. The original dataset has a big hole in the bottom of the head. We have manually filled in the hole since our model currently can only handle closed shapes. The input dataset of Fig. 5 is obtained by sampling a subdivision surface with 6140 data points.

Results from 2D Multiple Views: Fig. 6(b) is the reconstruction from a synthesized torus, demonstrating topology changes while recovering shape. The original dimensions of the synthesized torus are: $206 \times 205 \times 57$. Compared to ground truth, we get min error of 0.166, max error of 14.231, mean error of 2.544 and RMS of 1.087. Fig. 6(d) is the reconstruction of a mug from 16 real images that proves the ability of recovering the underlying topology from 2-D images. The use of local adaptive refinement on the top allows the model to get the detail of the small grip on the lid of the mug. The multi-resolution procedure has two levels of detail in Fig. 7. Fig. 7(b) is the result after adaptive refinement.

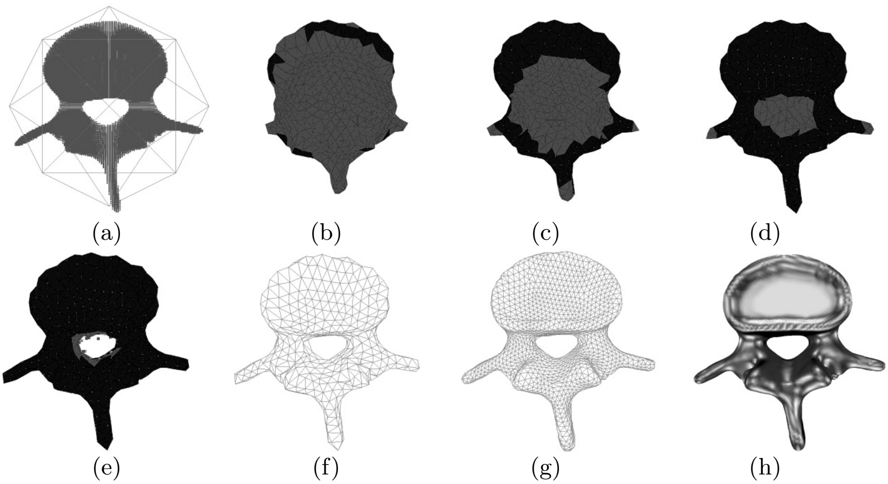


Fig. 3. Segmentation of a CT volumetric dataset of a human vertebra. (a) Initial model and input data; (b)-(e) model evolving; (f) mesh model; (g) optimized mesh model; (h) shaded model.

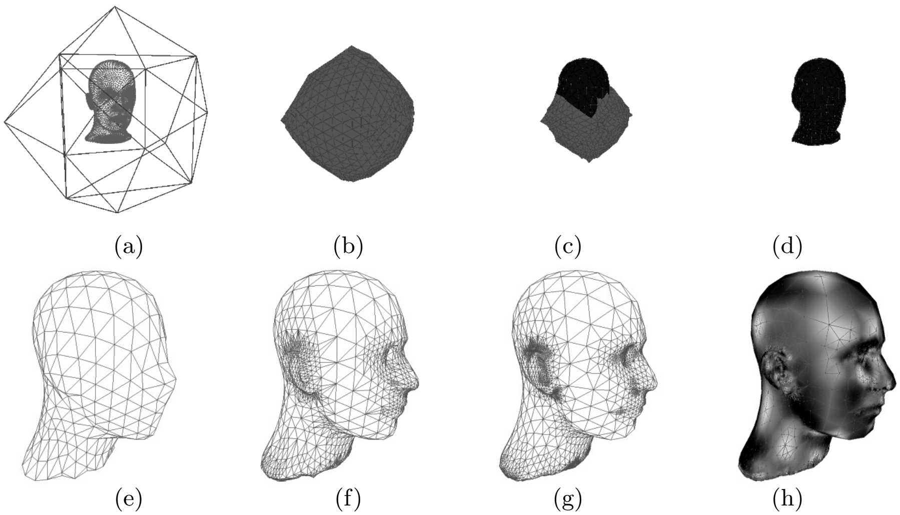


Fig. 4. Reconstruction from point-clouds data of mannequin head. (a) model initialization; (b) and (c) during deformation; (d)(e) final shape of the model; (f)(g) are two different levels of adaptive refinement; (h) is shaded result

It increases the resolution while avoiding global subdivision. Fig. 8(a) shows the positions of 16 real images of a Buddha figure. The 2 images with question marks (one of which is shown in 8(b)) were not used in the reconstruction. Fig. 8(c) is the final texture mapped mesh rendered from a similar viewpoint as the image in 8(b). Fig. 9 shows an example of incremental reconstruction. Fig. 9(a) is the

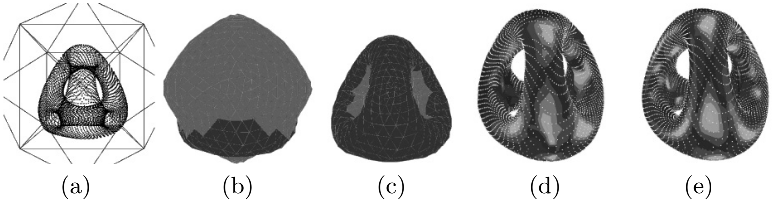


Fig. 5. (a) model initialization; (b) (c) during deformation, dark area stands for non-active, while grey is active; (d) final shape; (e) one level of refinement

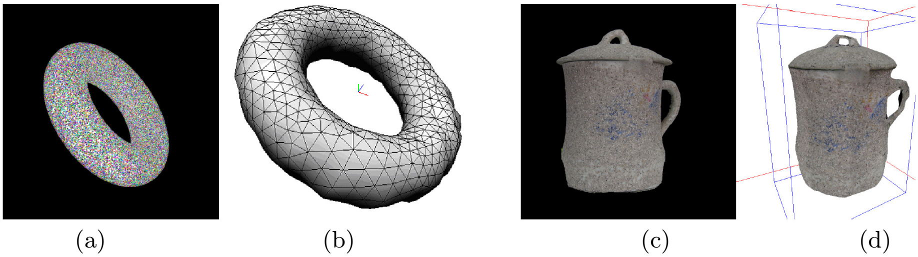


Fig. 6. Fitting to synthesized images with topology changes (a) one of 24 synthetic images of torus; (b) final mesh with 897 vertices after fitting to images (mean error of 2.544); (c) one of 16 real image of a mug; (d) final textured mesh after fitting to the images, the fine detail at the top is the result of local adaptive refinement.

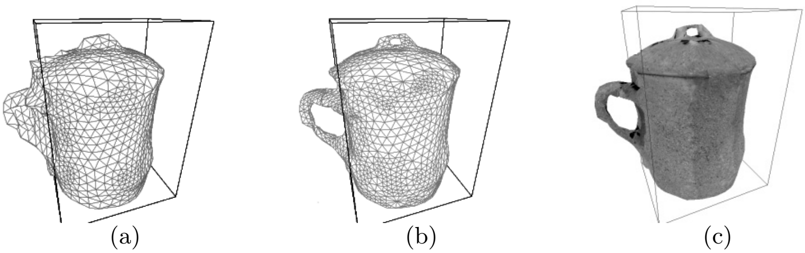
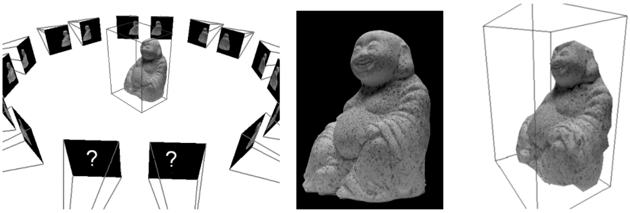


Fig. 7. Adaptive reconstruction. (a) initial reconstruction; (b) reconstruction after one level of adaptive refinement, the two handles of the cup are correctly recovered. (c) textured results from (b)

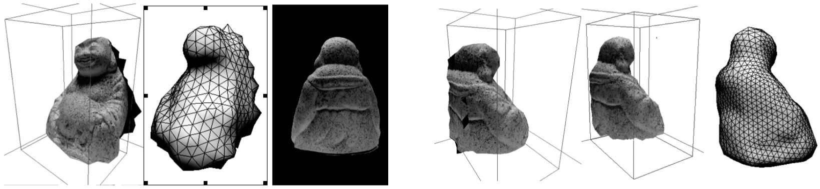
reconstruction result from 6 frontal images, and 9(b) is its mesh representation. The back of the model has not deformed due to the lack of image data. 9(c) is one of the 5 images added later. After adding images, the model further deforms in 9(d), and finally captures the complete shape shown in 9(e) and 9(f).

Table 1 gives the information of the recovered shape, including the number of vertices, edges and faces for each model, and the running time. The running time is measured on an Intel Pentium 4M 1.6GHZ Notebook PC with 384MB internal memory.



(a) (b) (c)

Fig. 8. (a) position of 16 real images of Buddha, the 2 images with question marks were not used in the reconstruction, one of them is shown in (b); (c) final textured mapped mesh rendered from a similar view point as images (b).



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

Fig. 9. Incremental reconstruction. (a): partial reconstruction result from 6 frontal images (Black area is not recovered due to lack of image of the data for that part of the model); (b): untextured fitting mesh of (a); (c): one of the 5 images (back views) added later; (d): intermediate result after adding more images; (e) complete reconstruction result; (f) mesh result of (e).

Table 1. Recovered shape information

Figure	Vertices	Faces	Edges	Time (sec)	Figure	Vertices	Faces	Edges	Time (sec)
4(e)	466	928	1392	23	5(d)	723	4774	7161	18
4(f)	3501	6998	10497	19	5(e)	2895	8688	5792	9
4(g)	8386	16768	25152	119	7(a)	1513	3018	4527	223
9(f)	1734	3464	5196	191	7(b)	2770	5540	8310	584

5 Discussion

In this paper, we proposed a new PDE-based deformable surface that is capable of automatically evolving its shape to capture geometric boundaries and simultaneously discover their underlying topological structure. The deformation behavior of the model is governed by partial differential equations, that are derived by the principle of variational analysis. The model ensures regularity and stability, and it can accurately represent sharp features. We have applied our model to shape reconstruction from volumetric data, unorganized point clouds and multi-view 2D images. The characteristics of the model make it especially useful in recovering 3D shape out of 2D multi-view images, handling visibility, occlusion and topology changes explicitly. The existence of a mesh representation allows progressive refinement of the model when appropriate. Our mathematical formulation allows us to use the same model for different types of data, simply by using the appropriate data interface function. We plan to further exploit this property in future work to apply the model to heterogeneous data such as points, surfels, images and to incorporate other visual cues such as shading and optical flow.

Acknowledgements. This work was partially supported by DOE grant 068, NSF-ITR/NGS grant ACI-0313184, NSF ITR grant IIS-0082035, the NSF grant IIS-0097646, the NSF grant ACI-0328930, the ITR grant IIS-0326388, and Alfred P. Sloan Fellowship. We would like to thank Professor Tim McInerney from University of Toronto for the vertebra dataset and Dr. Hughes Hoppe from Microsoft Research for the mannequin head dataset, respectively.

References

1. N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. *SIGGRAPH*, pages 415–421, 1998.
2. P. Besl and H. McKay. A method for registration of 3d shapes. *PAMI*, 1992.
3. J.S. De Bonet and P. Viola. Poxels. Probabilistic voxelized volume reconstruction. *ICCV*, pages 418–425, 1999.
4. J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *SIGGRAPH*, 2001.
5. V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces based object segmentation. *PAMI*, 19, 1997.
6. Charles. Smooth subdivision surfaces based on triangles. Master's thesis, Mathematics, Univ. of Utah, August 1987.
7. B. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. *In International Workshop on Vision Algorithms*, 1999.
8. D. DeCarlo and D.N. Metaxas. Blended deformable models. *PAMI*, 1996.
9. R. Deriche, C. Bouvin, and O. D. Faugeras. Front propagation and level-set approach for geodesic active stereovision. *ACCV*, 1:640–647, 1998.
10. H. Q. Dinh, G. Slabaugh, and G. Turk. Reconstructing surfaces using anisotropic basis functions. *ICCV*, 2001.
11. H. Edelsbrunner and E.P. Mucke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13:43–72, 1994.
12. O.D. Faugeras and R. Keriven. Complete dense stereovision using level set methods. *ECCV*, pages 379–393, 1998.
13. P. Fua. From multiple stereo views to multiple 3d surfaces. *IJCV*, 24:19–35, 1997.
14. H.K.Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. *VLSM Workshop*, July 2001.
15. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH*, 1992.
16. H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo beyond lambert. *CVPR*, 2003.
17. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, pages 321–331, 1988.

18. R. Kimmel. 3d shape reconstruction from autostereograms and stereo. *Journal of Visual Communication and Image Representation*, 13:324–333, 2002.
19. R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. In *Proceedings of National Academy of Sciences*, pages 8431–8435, July 1998.
20. L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Eurographics*, pages 249–260, 2000.
21. K.N. Kutulakos and S.M. Seitz. A theory of shape by space carving. *ICCV*, 1999.
22. J.-O. Lachaud and A. Montanvert. Deformable meshes with automated topology changes for coarse-to-fine 3d surface extraction. *Medical Image Analysis*, 3(2):187–207, 1999.
23. W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1997.
24. R. Malladi, J. Sethian, and B. Vemuri. Shape modeling with front propagation: A level set approach. *PAMI*, 17(2), 1995.
25. T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2), 1996.
26. T. McInerney and D. Terzopoulos. Topology adaptive deformable surfaces for medical image volume segmentation. *TMI*, pages 840–850, 1999.
27. D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *PAMI*, 15(6), 1993.
28. J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara, and M.J. Wozny. Geometric deformed models: a method for extracting closed geometric models from volume data. *SIGGRAPH*, pages 217–226, 1991.
29. M. Pollefeys and L. J. Van Gool. From images to 3d models. *CACM*, 7:50–55, 2002.
30. S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. *Proceedings of 3D Digital Imaging and Modeling*, pages 145–152, 2001.
31. Szymon Rusinkiewicz, Olaf A. Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. *ACM Transactions on Graphics*, 21(3):438–446, 2002.
32. S.M. Seitz and C.M. Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 35(2):1–23, 1999.
33. Gregory G. Slabaugh, Ronald W. Schafer, and Mat C. Hans. Multi-resolution space carving using level sets methods. *ICIP*, 2002.
34. C.J. Taylor and D. Jelinek. Structure and motion from line segments in multiple images. *PAMI*, 17(11), 1995.
35. E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vsion.* Prentice Hall, 1998.
36. Z. Wood, M. Desbrun, P. Schroder, and D. Breen. Semi-regular mesh extraction from volumes. In *Proceedings of IEEE Visualization*, pages 275–282, 2000.