

which is caused by a small narrow ridge section (about 8mm thick, much smaller than 2 times R_f , where $R_f = 10\text{mm}$). We believe that this can be solved by examining and identifying local surface changes more closely and adjusting system parameters accordingly in that area.

10 Conclusions and future research

We have presented a surface description method based on a dynamic balloon model using a triangular mesh with springs attached to the vertices. The balloon model is driven by an applied inflation force towards the object surface from inside of the object, until all the triangles are anchored on the surface. The model is a physically based dynamic model and the implementation of the algorithm is highly parallelizable. Furthermore, our system is not based on global minimization and can allow the model to adapt to local surface shapes based on local measurements. Experiments showed very good results on complex, nonstar-shaped object. Notice that there are still many improvements that can readily be made on the resulting model, including using the algorithms presented in [Chen & Medioni 93] to improve triangle fitting errors, or the method in [Soucy & Laurendeau 92] to merge small triangles into larger ones without affecting the fitting error for constructing a hierarchical representation. Local smooth patches can also be constructed for high level surface property analysis. In addition, our future research will consist of detecting and avoiding possible self-intersections of the mesh surface.

References

- [Chen & Medioni 92] Y. Chen and G. Medioni, "Object Modelling by Registration of Multiple Range Images," *International Journal of Image and Vision Computing (IVC)*, 10(3):145–155, April 1992.
- [Chen & Medioni 93] Y. Chen and G. Medioni, "Surface Level Integration of Multiple Range Images", in the *Proceedings of the Workshop on Computer Vision for Space Applications*, Antibes, France, September 1993.
- [Cohen & Cohen 93] Laurent D. Cohen and Isaac Cohen, Finite-element methods for active contour models and balloons for 2-D and 3-D images, *IEEE Trans.* Vol. PAMI 15, 1993, pp.1131-1147.
- [Delingette et al. 91] H. Delingette, M. Hebert, K. Ikeuchi, "Shape Representation and Image Segmentation Using Deformable Surfaces," *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.467-472, Maui, HI, June 1991.
- [McInerney & Terzopoulos 93] T. McInerney and D. Terzopoulos, "A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking", *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 518-523, Berlin, Germany, May 1993.
- [Rivara 84] M. Cecilia Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques", *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 745-756, 1984.

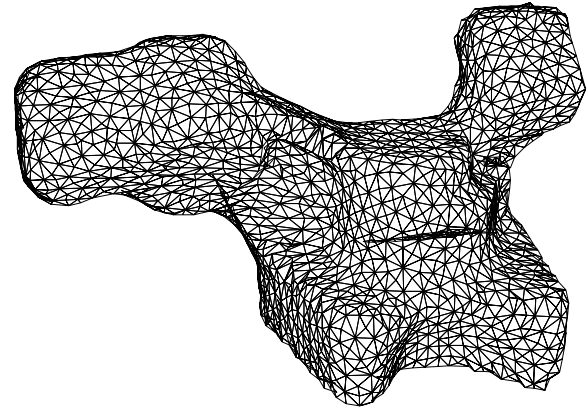
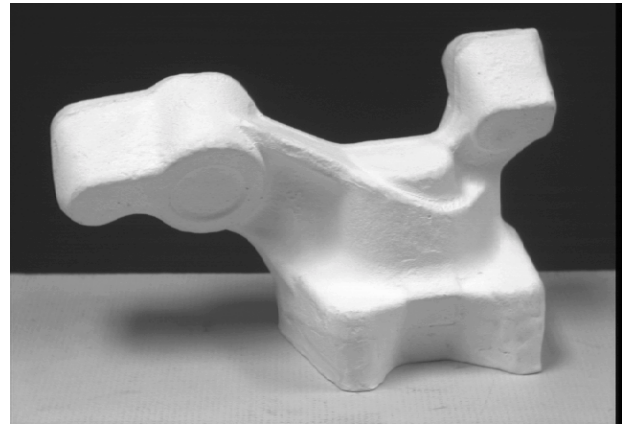


Figure 9. Modeling the Renault part. Top: intensity of the original object. Middle: wireframe drawing. Bottom: rendered image of the constructed model.

- [Sato & Inokuchi 87] K. Sato and S. Inokuchi, "Range-Imaging System Utilizing Nematic Liquid Crystal Mask," *ICCV*, pages 657–661, London, England, June 1987.
- [Soucy & Laurendeau 92] M. Soucy and D. Laurendeau, "Multi-resolution surface modeling from range views," *CVPR*, pp.348–353, Urbana-Champaign, IL, June 1992.
- [Terzopoulos & Vasilescu 91] D. Terzopoulos and M. Vasilescu, "Sampling and Reconstruction with Adaptive Meshes" *CVPR*, pp.70-75, Maui, HI, June 1991.
- [Vasilescu & Terzopoulos 92] M. Vasilescu and Demetri Terzopoulos, "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision," *CVPR*, pp. 829-832. Urbana-Champaign, IL, June 1992.

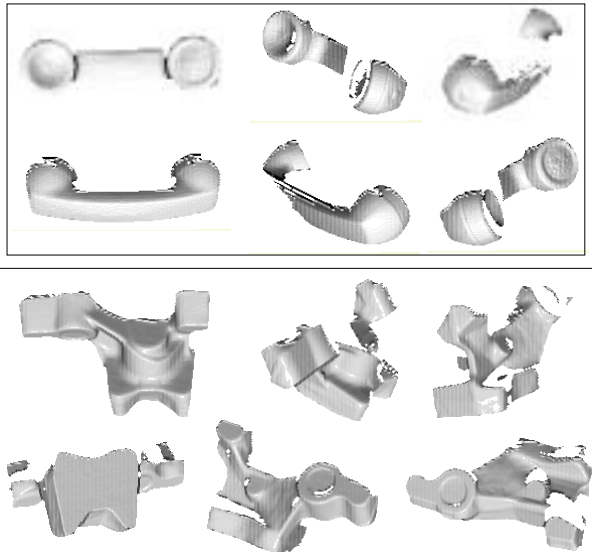


Figure 5. Sample range images used in constructing the Phone model and Renault model in Figure 8 and Figure 9, shown here as shaded intensity images.

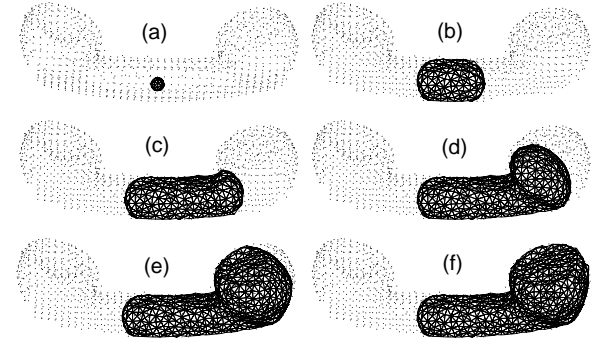


Figure 6. Stages of an inflating balloon inside the Phone, showing the movement of the right front only. The wireframes are superimposed with sample points from the input range images.

ters can be found in section 7. Both the Phone and the Renault part measure about $200mm$ across their longest sides. Note that the wireframe drawings presented here are *not* produced using a true hidden-line elimination algorithm, which is the cause of most of the spurious triangles seen in the wireframe drawings.

As can be seen from the results presented above, our algorithm works very well on both simple, compact objects and nonstar-shaped objects with complex structures. The resulting triangulated model surfaces preserve most of the important geometry feature of the objects with evenly distributed meshes. Our initial meshes are all set in the neighborhood of the center of the objects measuring $2mm$ and yet our balloons can successfully grow to cover all parts of the object with complex geometric structures like the Renault part. Also, despite of our effort to acquire many range image views to cover the whole object, the data that we use for the Renault part contain holes on top of both arms, and the resulting

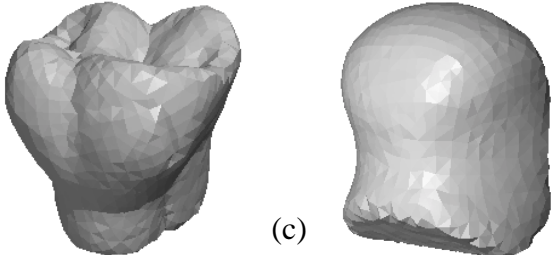
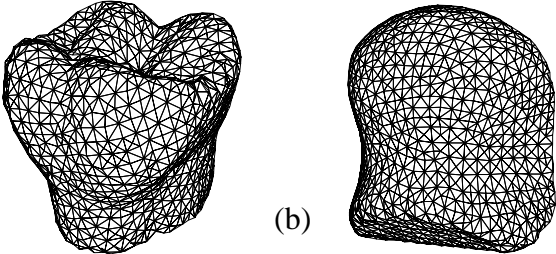
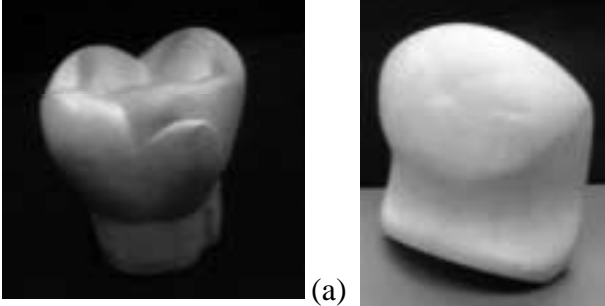


Figure 7. Examples of the balloon model for fitting simple objects: (a) the original intensity images of the objects, (b) the wireframes of the obtained balloon models and (c) the rendered shaded images of the models.

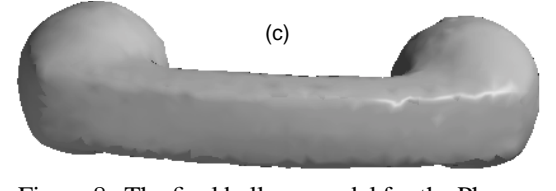
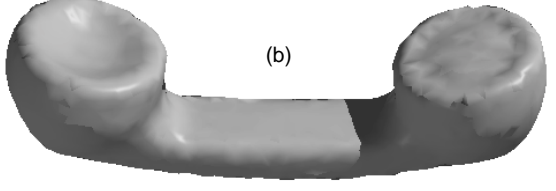
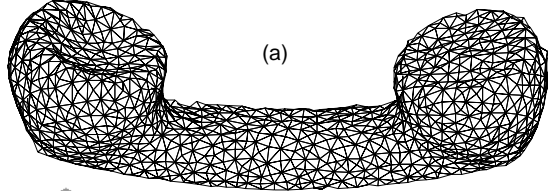


Figure 8. The final balloon model for the Phone: (a) wireframe (b) and (c) smoothly shaded.

mesh was able to interpolate them very well. There is, however, a defect under the right arm of the Renault part, as can be seen in the wireframe drawing. It is a small opening in the mesh that tends to self-intersect

is to approximate the object surface to have a triangle fitting error δ for surfaces with maximum curvature of $1/R_t$, S_t can be easily computed by tessellating a unit sphere of radius R_t with equilateral (or near equilateral) triangles of sizes smaller or equal to S_t . This also gives us a sample configuration of an ideal front structure when the maximum mesh tension is achieved. Let $f_{spring-max}$ be the maximum spring force exerted on to a vertex under such conditions. The inflation force needed to overcome the spring force (in order for the vertices to move) is therefore

$$f_{inflate} > f_{spring-max} \quad (7)$$

The inflation force is also constrained by equation (3), since once a time step and a maximum displacement per iteration are set, the allowed inflation force should then be

$$f_{inflate} \leq \frac{d_{max}}{\Delta t} + g \quad (8)$$

where $d_{max} = \max(\|x^{t+\Delta t} - x^t\|)$ is the maximum displacement. Since a large inflation force tends to dominate the mesh's evolution, which we don't want, we prefer a smaller one. We choose to use the minimal inflation force as shown in equation (7). We can then compute the needed inflation force amplitude k according to equation (5).

Now the whole issue comes down to determining $f_{spring-max}$, d_{max} and the time step Δt . The maximum spring force is determined by the spring natural length l_{ij} and spring stiffness which are related (equation (4)). In our experiments, we have used $l_{ij} = 0$ and $c_{ij} = 4.0$. d_{max} and Δt are selected to allow the mesh to evolve smoothly and quickly relative to the object size and complexity. For all the tests in this paper, we have used $2mm$ and 0.05 respectively.

Finally, the user needs to select δ and R_t . For the purpose of simplicity, in our experiments, however, we manually set R_t and allow a fixed number of N triangles to fit the sphere with a radius R_t , which gives a nominal approximation error of about $0.6mm$ with $N = 80$ and $R_t = 10mm$. These parameters also apply to all the experiments in this paper.

8 Dealing with holes and noise

It is worth mentioning that our algorithm presented in section 6 is parallelizable since the computations on each front in the queue Q are independent of each other. Furthermore, the computation for each vertex within a front is also independent at each iteration.

Another advantage that this computation structure brings us is that we can adaptively adjust system parameters independently for each front based on the information that we gather from the prospective correspondence

points of the vertices in the front. For example, if we have detected that the movement of the front is virtually stopped and yet the prospective correspondence points are still certain distance away, this tells us that the preset parameter R_t in previous section is too large and we should adjust it accordingly.

Another example of such adaptation is in handling holes in data. In this case, there exist areas of the object surface that are not covered by any of the input range images, we will not be able to find prospective correspondence points for some of the vertices in the related front. Eventually, when the rest of the vertices in the front have settled down to their correspondence points, we are left with a front for which none of the vertices have a prospective correspondence point. In such situations, the system automatically sets the inflation force to zero ($k = 0$ in Equation [5]), which allows the mesh to reach an equilibrium state that interpolates the surface over the hole.

Another important issue is the issue of noise. There are two type of noises that may affect our results. One is the noise introduced by the small misalignment among the range images. The other is the outliers from each range image. Our system is very stable with respect to both types of noise. The first one is effectively solved by the weighted sum line-surface intersection algorithm (section 3.1) since the misalignment causes the actual intersections to form a cluster. The second type of noise usually cause the intersection algorithm on the related range image to fail to converge, in which case it does not contribute to the result of the intersection. Even if the noise does produce a wrong intersection, it can easily be filtered out as an outlier that does not belong to the correct cluster.

9 Test results and discussion

We now present some examples of our balloon model in modeling a model telephone handset (Phone) and an automobile part (Renault) using 20 and 24 range image views respectively. The range images are acquired using a Liquid Crystal Range Finder (LCRF) [Sato & Inokuchi 87], and then registered using the range image registration algorithm described in [Chen & Medioni 92]. Some sample range images used in the experiments are shown in Figure 5. Figure 6 shows a growing balloon inside the Phone at different stages of the growth. Note that our algorithm works on the fronts in a sequential order so the figures only show the process of the right front. In Figure 7, examples for some simple and compact objects are shown. In Figure 8, the final rendered views of the constructed model of the Phone are shown below the wireframe drawing. Figure 9 shows a wireframe and the rendered image of the Renault part. The final model for the Phone has 1694 vertices and 3384 triangles, the Renault part has 2850 vertices and 5696 triangles. The used system param-

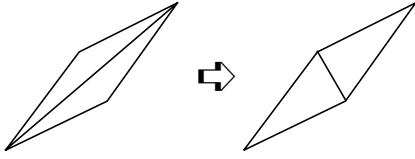


Figure 4. Rearranging the local configuration to eliminate long and thin triangles.

angle is lower-bounded by half of the smallest inner angle of the original triangles [Rivara 84].

This algorithm, however, does not guarantee that the triangles on the boundary areas of a front conform with the rest of the triangles in the triangulation. Hence, after the algorithm is finished, we must bisect the affected non-conforming triangles accordingly. Thus we have:

Algorithm 2

- 1) Carry out Algorithm 1 on the set of triangles $\tau_0 \subset \tau_{f_0}$.
- 2) For each non-conforming triangle $T \subset \tau_{f_0}$ but connected to τ_{f_0} , bisect T by its non-conforming edge.

An example of this algorithm is shown in Figure 3,

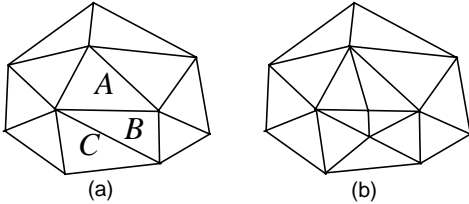


Figure 3. Subdivision of triangle mesh: (a) before A is subdivided, (b) after A is subdivided and subdivision is propagated to both B and C.

where triangle A is to be subdivided and C does not belong to the region (front). As can be seen from the figure, the subdivision is propagated to B and finally C is bisected to make the triangles at the region boundary conforming (step 2 above).

5.2 Local mesh adjustment

The above algorithm works very well under most circumstances, but degenerate triangles that are long and thin may still occur. These triangles are undesirable since they do not represent local surface shape well and are often the cause of self-intersection of the mesh surface. Currently, we use a simple algorithm that checks for pairs of such triangles and rearrange the triangle configuration locally. After each subdivision, we check for triangles that are thin and long, and if two such triangles share an edge that is the longest for both triangles, then we simply switch the cross edge as shown in Figure 4.

6 Description of the algorithm

In this section we give a brief description of the entire algorithm of our approach. A discussion on how to set the system parameters will follow. We assume that registered range image views of the object to be modeled are available, although we believe the algorithm can be adapted to other types of 3-D input.

We start with selecting an initial point inside the object and constructing an icosahedron shell [Delingette et al. 91] at this location. The selection process is currently done by hand and the size of the shell should be small enough so that it is completely inside the object. The system algorithm can be described as follows.

Algorithm 3

Let all the triangles on the initial mesh be front F_0 and push it into the front queue Q , then until queue is empty, do the followings repeatedly:

- 1) $F \leftarrow$ top of the queue Q , pop the queue.
- 2) Subdivide the triangles in F if appropriate (see next section.)
- 3) For each vertex $V_i \in F$, whose 3-D coordinates at time t is \mathbf{v}_i^t , do
 - a) compute the internal force \mathbf{g}_i and external force $\mathbf{f}_i = \mathbf{h}_i$ based on equations (4) and (5).
 - b) compute the new vertex location $\mathbf{v}_i^{t+\Delta t}$ for the current iteration according to equation (3).
 - c) compute prospective correspondence point of \mathbf{v}_i , which is the intersection \mathbf{w}_i of the surface and the ray through \mathbf{v}_i and in the direction of the mesh normal at \mathbf{v}_i (section 3.1.)
 - d) if $\|\mathbf{v}_i^{t+\Delta t} - \mathbf{v}_i^t\| > \|\mathbf{w}_i - \mathbf{v}_i^t\|$, then $\mathbf{v}_i^{t+\Delta t} \leftarrow \mathbf{w}_i$ and mark vertex V_i anchored.
- 4) For each $V_i \in F$, update its position with the corresponding new positions $\mathbf{v}_i^{t+\Delta t}$.
- 5) Discard triangles from F that have thus become anchored (section 5).
- 6) if $F = \{\emptyset\}$ then go to 1.
- 7) recompute connected triangle regions in F and push them into Q . Go to 1.

The intersection point \mathbf{w}_i in step (3)(c) is the closest intersection to the mesh surface. If there exist multiple intersections for the ray with many range image views, the median of the closest cluster of the intersections is used.

7 Setting up the parameters

In our current implementation, triangles that have areas larger than S_t are subdivided at each iteration. S_t is directly related to the precision of the fit of the final mesh to the input surface data. Assuming that our goal

The inflation force a vertex receives takes the form of:

$$\mathbf{h}_i = k\hat{\mathbf{n}}_i \quad (5)$$

where k is the amplitude of the force and $\hat{\mathbf{n}}_i$ is the direction normal to the local model surface. In our implementation, the normal at a triangle vertex is estimated from the vector sum of the normal vectors of the surrounding triangles:

$$\hat{\mathbf{n}}_i = \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}, \quad \mathbf{n}_i = \sum \frac{(\mathbf{n}_{ij} + \mathbf{n}'_{ij})}{\|(\mathbf{n}_{ij} + \mathbf{n}'_{ij})\|}. \quad (6)$$

where \mathbf{n}_{ij} is the direction normal to the j th triangle $j \in \{T_j\}$ with $\{T_j\}$ being the triangles surrounding the vertex, and \mathbf{n}'_{ij} is the direction normal to triangle T'_j that is the neighbor to T_j but $j' \notin \{T_j\}$. This estimation is more stable than what we get when only the triangles in $\{T_j\}$ are used.

5 Mesh subdivision and adaptation

In our simulated physical system, during the process of the growth of the mesh model, the mesh triangles become bigger in size, and tensions due to the spring force also build up, which will eventually stop the movement of the mesh, and the inflation force and the spring tension enter an equilibrium state. This is not desirable since we do not consider forces from the input data, so an equilibrium state does not mean a good fit. In order to keep the balloon growing, we can keep the inflation force unchanged (which actually means to keep on inflating) and at the same time reduce the spring tension by subdividing the triangles in the mesh into smaller triangles. Alternatively, we can increase the inflation force and allow the spring tension to increase. But increasing the inflation force also has the side effect of increasing the maximum displacement. As can be seen from equation (3), the spring force \mathbf{g}_i acts as a balance force to the inflation force $\mathbf{f}_i = \mathbf{h}_i$ (assuming a convex local structure), thus the maximum displacement is directly related to the inflation force once a time step is chosen. So we choose not to increase the inflation force, but to subdivide the mesh instead.

The purpose of subdividing triangles is twofold. Once a triangle is subdivided, the sides of the triangles becomes shorter and if we keep the natural length and stiffness of the springs constant, the spring tension is reduced. Also, subdividing the triangles helps maintain an evenly distributed mesh. Subdividing triangles in certain region, as will be discussed later, also allows the mesh surface to adapt to the local object surface geometry without affecting other parts of the mesh surface.

Before introducing the details of the triangle subdivision, we first define some terms. A vertex is said to be *anchored* if it has reached the object surface and has been marked as such. A triangle is said to be anchored if all of its vertices are anchored. At any time in the mesh growing process, the triangles in the mesh can be classified into anchored triangle regions, consisting of anchored triangles, and unanchored triangle regions, consisting of movable triangles called *front*. Each front is a connected component of triangles, in which two triangles are said to be connected iff they share an edge.

5.1 Adaptive triangle mesh subdivision

Triangle subdivision is carried out only on the front, since anchored triangles are not allowed to move. This allows the triangular mesh to adapt to the object surface better without globally adjusting the position of all vertices. A good subdivision scheme would be one that yields an evenly distributed mesh and produces few degenerate (i.e. long and thin) triangles. The algorithm that we use in this paper first selects a set of triangles that needs to be subdivided through bisection. Then after these triangles are bisected on their longest edges, adjacent triangles are also bisected or trisected to make the triangles *conforming*, the state that a pair of neighboring triangles either meet at the a vertex or share an entire edge. In our implementation, only those triangles that exceed certain size limit are subdivided first. The algorithm presented below is adapted from Algorithm 2 (local) in [Rivara 84], which is developed for refining triangular meshes for finite element analysis.

Algorithm 1

Let τ_{f0} be the set of the triangles from a given front, and $\tau_0 \subset \tau_{f0}$ is the selected set of triangles to be subdivided.

- 1) Bisect T by its longest edge, for each $T \in \tau_0$.
- 2) Find $R_1 \subset \tau_0$ the set of non-conforming triangles generated in step 1. Set $k \leftarrow 1$.
- 3) For each $T \in R_k$ with non-conforming point $P \in T$ (mid-point on the non-conforming edge): (a) bisect T by the longest edge; (b) if P is not on the longest edge of the T , then join P with the midpoint of the longest edge.
- 4) Let τ_0^k be the triangulation generated in step 3. Find $R_{k+1} \subset \tau_0^k$ the set of non-conforming triangles generated in step 3.
- 5) If $R_{k+1} = \{0\}$, stop, the subdivision is done. Else, set $k \leftarrow k + 1$ and go to step 3.

This subdivision algorithm has the feature that the subdivision will only be propagated towards large triangles from the longest edge of a subdivided triangle. It is also proven that the resulting triangles' smallest inner

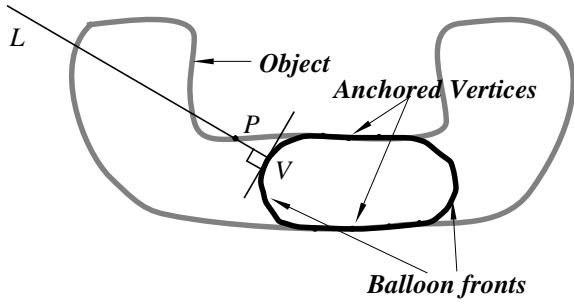


Figure 2. Line-surface intersection: searching for a corresponding point in the normal direction.

the distance from the data points on the surface to the nearest model point is used instead, which is an improvement over the above approach. This approach, however, is not practical when there is a large number of surface sample points from the object, as in our case.

In our approach, we “probe” for potential corresponding points only in the direction normal to the mesh surface. This is the best knowledge locally available to the points on the mesh surface at any time during the mesh’s growing process, because the equivalent mesh surface movement in the neighborhood of a mesh element is only in the direction normal to the mesh surface. So it is only natural to look for corresponding point from the object surface in the direction of the normal, which will also change in the process of inflation.

In our implementation, the “probing” is performed by computing the prospective corresponding point P (Figure 2), the closest intersection of a ray in the normal direction and the object surface represented in dense range images. Details of the basic algorithm can be found in [Chen & Medioni 92]. A brief description is given below. The line-surface intersection algorithm works directly on range images without an analytical representation of the surface. At each iteration, the object surface near the prospective intersection point is approximated linearly using its tangent plane, which intersects with the line in consideration. This intersection converges to the intersection point of the surface and the line. When there are intersections between the line and more than one range images, a weighted sum of all those intersections is used as the intersection. Once the intersection is found, the distance from the mesh surface to the intersection can be used as a measure of whether the mesh has reached the object surface.

When there are holes in the input data (parts of the object surface not covered by the input data), we will not be able to find the intersections described above. In such cases, there are no prospective correspondence points for the affected vertices and thus there is no reason to apply inflation force further more. This kind of decision, however, should not be made locally for each

vertex point. We will talk more about handling holes in section 8.

4 A simplified dynamic model

The motion of any element i on the model surface can be described by the following motion equation [Terzopoulos & Vasilescu 91]:

$$m_i \ddot{x}_i + r_i \dot{x}_i + g_i = f_i, \quad i = 1 \dots N \quad (1)$$

where x_i is the location of the element, \dot{x}_i and \ddot{x}_i are the first and second derivatives with respect to time, m_i represents the mass, r_i is the damping coefficient, g_i is the sum of internal forces from neighboring elements due to, e.g., spring attachments and f_i is the external force exerted on the element. Because of the nonlinear nature of the forces g_i and f_i involved, the systems of ordinary differential equations (1) can be solved using explicit numerical integration [Terzopoulos & Vasilescu 91].

The dynamic system will reach the equilibrium state when both \dot{x}_i and \ddot{x}_i become 0, which can take a very long time since it is usually an exponential process. A simplified system can be obtained if we make $m_i = 0$, and $r_i = 1$ for all i , in which case equation (1) reduces to

$$\dot{x} = f_i - g_i, \quad i = 1 \dots N \quad (2)$$

The reason for making this simplification is that our goal for the dynamic system is for the elements in the mesh to reach the object surface and we will stop them without considering whether the system is in an equilibrium state or not. We also do not intend to have a special treatment for any elements in the mesh at this time, so all r_i should be equal, in which case we can normalize the parameters so that $r_i = 1$. The set of first-order differential equations in equation (2) has a very simple explicit integration form as follows:

$$x^{t+\Delta t} = (f_i^t - g_i^t) \Delta t + x^t \quad (3)$$

4.1 Spring force and inflation force

The spring force exerted on vertex i by the spring linking vertex i and j can be expressed as [Terzopoulos & Vasilescu 91]:

$$s_{ij} = \frac{c_{ij} e_{ij}}{\|r_{ij}\|} r_{ij} \quad (4)$$

where c_{ij} is the stiffness of the spring, $e_{ij} = \|r_{ij}\| - l_{ij}$ is the spring deformation, $r_{ij} = x_j - x_i$, $\| \cdot \|$ is the vector length operator and l_{ij} is the natural length of the spring. The total spring force g_i a vertex receives is the vector sum of spring forces from all springs attached to it.

Another aspect of our system is its role in data integration. Most of the previous research make use of *scattered* 3-D points or a *single range image* as input. Very few (e.g. [Soucy & Laurendeau 92]) try to use *multiple* densely sampled range images. There are two difficulties in using these range images. First, the images must be precisely registered, which is solved by using a range image registration method we published previously [Chen & Medioni 92]. The second is the issue of representation. Integration can not be done without an appropriate representation for the integrated data. For nonstar-shaped objects, it is more difficult since the surface points of an object can not be simply mapped onto a unit sphere, which would be sufficient for star-shaped objects [Chen & Medioni 92]. Thus the choice of representation is very important. This, however, is not the theme of this paper.

In the following sections, we first review some of the related previous work and then present our balloon model in detail. Section 3 describes the model and how it works, section 4 defines the dynamics of the system. Section 5 explains the adaptive mesh subdivision scheme. In section 6, we give an algorithmic description of our system. Section 7 and 8 discuss how the system parameters are set and how to adapt the parameters locally. We present our test results in section 9 and our conclusions in section 10.

2 Related work

Dynamic mesh models have been proposed by previous researchers for shape description. Terzopoulos and Vasilescu [Terzopoulos & Vasilescu 91] introduced a shape reconstruction algorithm using a dynamic mesh that can dynamically adjust its parameters to adapt to the input data. They also extended this approach by introducing an attraction force from 3-D inputs for shape description [Vasilescu & Terzopoulos 92]. Delingette *et al.* [Delingette et al. 91] proposed a deformable model with internal smoothness energy and external forces from both the input data and features. There are other deformable model approaches that differ in representation schemes of the model and in the approaches to solving the system (see [Cohen & Cohen 93] and [McInerney & Terzopoulos 93]).

The main difference between our method and those used by previous researchers is that we do not explicitly introduce a data force into our model, as will be discussed in detail in the following sections. Our model is purely driven by an inflation force introduced inside the balloon. Balloon models have been used by [Cohen & Cohen 93] and [McInerney & Terzopoulos 93], but in their approaches, the introduced balloon force is used to assist the global energy minimization model to converge to the desired results and to overcome some noise in the volume data, and a careful balance must be maintained between the balloon force and other forces.

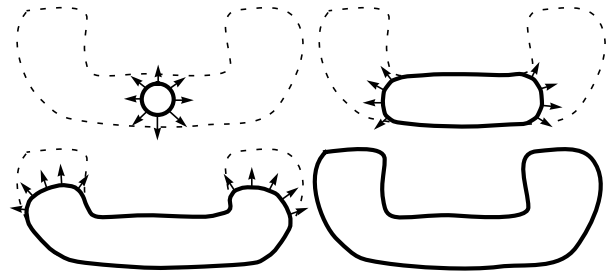


Figure 1. The inflating balloon model as illustrated in a 2-D case (from left to right and from top to bottom): the initial state, intermediate states and the final state.

3 The inflating balloon model

Our balloon model is represented by a shell of triangulated patches. The initial triangulated shell is an icosahedron. When placed inside the object and under the inflation force applied to the vertices, the shell grows in size as the vertices move along the mesh surface normal in the radial direction, maintaining a spherical shape, until one or more vertices reaches the object surface. During the process of inflation, the triangles may be subdivided adaptively, which also creates new vertices. Once it reaches the surface, a vertex is considered fixed to the surface and thus can no longer move freely. The remaining vertices will keep moving until reaching the surface (Figure 1). As can be seen from this process, the movement of the vertices is not influenced by any force from the surface of the object. This seems to be bad from the point of view of a fitting process, which tries to minimize some distance measure between the object surface and the model. But this is important in our case because our main concern is to find a correct mapping between the mesh elements and the object surface. By not using any attraction force from the surface data it allows us to avoid incorrect mappings (which is a similar situation to the local minimum in energy minimization approach) without depending on a very close initial surface.

3.1 The correspondence problem

So far we have not discussed how to test whether the mesh has reached the surface of the object. This is the key difference between our approach and other dynamic model systems or energy minimization systems. In order to test whether a vertex has reached the object surface, one must measure the distance between the mesh surface and some point on the object surface. Ideally this point on the object surface should be the corresponding point of the vertex, which is not possible before the vertex reaches the surface. Previous researchers have used the closest point on the object surface to a mesh element as an alternative, but it may provide incorrect information. In [McInerney & Terzopoulos 93]

Surface Description of Objects from Multiple Range Images*

Yang Chen and Gérard Medioni

Institute for Robotics and Intelligent Systems

University of Southern California

Los Angeles, California 90089-0273

E-mail: yangchen@iris.usc.edu

Abstract

We address the problem of constructing a complete surface model of an object using a set of registered range images. Our approach is based on a dynamic balloon model represented using a triangulated mesh. The vertices in the mesh are linked to their neighboring vertices through springs to simulate the surface tension and to keep the shell smooth. Unlike other dynamic models proposed by previous researchers, our balloon model is purely driven by an applied inflation force towards the object surface from inside of the object, until the mesh elements reach the object surface. The system includes an adaptive local triangle mesh subdivision scheme that results in an evenly distributed mesh. Since our approach is not based on global minimization, it can handle complex, non-star-shaped objects without relying on a carefully selected initial state or encountering local minimum problem. It also allows us to adapt the mesh surface to changes in local surface shapes and to handle holes present in the input data through adjusting certain system parameters adaptively and locally. We present results on some complex, nonstar-shaped objects as well as simple, compact objects from real range images.

1 Introduction

The task of surface description using 3-D input is can be described as fitting a chosen representation (model surface) to the input data points. This process can be formalized as the minimization of a system functional that represents the fit of the model to the input data explicitly or implicitly. Another very important aspect of such a system is to construct a *mapping* or *correspondence* between the surface of an object and the structure of the model. This mapping exists because the surface of the model and the surface of the object are topologically equivalent, considering genus zero type of objects. Therefore there exists a one-to-one mapping

between the model structures and the object surface elements. Previous researchers have studied such mappings in a variety ways using different representation schemes and model fitting methods. Examples of these approaches are the dynamic system using energy minimization of [Delingette et al. 91] and the dynamic mesh of [Terzopoulos & Vasilescu 91] and [Vasilescu & Terzopoulos 92]. One of the drawbacks of these approaches is that it is yet not known how to formulate the system in such a way that it is guaranteed to converge to the desired result, or in other words, they must rely on an initial guess of the model structure that is relatively close to the shape of the object. That is why those approaches can only deal with star-shaped objects in general.

In this paper we present a new approach for surface description using a dynamic balloon model represented by a triangular mesh. We start with a small triangulated shell placed inside the object and apply a uniform inflation force on all vertices in the directions normal to the shell's surface. The vertices are also linked to their neighboring vertices through springs to simulate the surface tension and to keep the shell smooth. The applied inflation force moves the vertices towards the object surface until they "land" on it. This process is similar to that of blowing up a balloon placed inside a hollow object until it fits the shell of the object. Thus the goal of mapping the model to the object surface is achieved through the physics of a growing balloon in a very natural way.

Our system is not based on global minimization methods, and it can make decisions based on local information about the shape of the object surface. As the mesh expands and the vertices start to reach the object surface, the entire mesh surface will gradually be subdivided into pieces of connected triangular regions, which allows us to treat the surface based on local surface properties by tailoring the parameters, and possibly strategy, of the system in dealing with each region separately. One such example is that it allows us to handle the case when there are holes in the input data.

* This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-90-C-0078 and/or Grant No F49620-93-1-0620. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.