FLEXIBLE SURFACE DEFORMATION FOR MESH EDITING

Kai Wang *Jianmin Zheng* *Hock Soon Seah*

School of Computer Engineering, Nanyang Technological University, Singapore 639798

ABSTRACT

Surface deformation techniques for triangular meshes have become one of the key topics in computer graphics and geometric modeling. It is often required for a deformation algorithm to present a natural and smooth deformation result while preserving the intrinsic geometry details of the mesh as much as possible. In this paper, we present a mesh deformation algorithm which strives to preserve the local shape of the mesh under three affine transformations: translation, rotation and scale. At the same time, flexibility is given to the users to balance between the rigidity and smoothness of the deformation by adjusting a weight parameter. A satisfactory result could be obtained by setting an appropriate weight value. Several experimental results are shown in this paper to demonstrate the effectiveness and flexibility of our algorithm.

Keywords: Computer graphics, geometric processing, surface deformation, mesh editing

1. INTRODUCTION

Surface deformation for triangular meshes has become a challenging problem in geometric processing and computer graphics. It is a fundamental process in mesh editing. One of the basic requirements for deformation is often that the local shape feature of the mesh should be preserved during deformation. Since translation and rotation will not change the geometric shape, in surface deformation the preservation of local geometric feature is meant up to some translation and rotation (or even scale in a more general sense) transformations applied on part or the whole of the model. This is quite complicated in three-dimensional space: additional degrees of freedom bring more issues that need to be tackled. In general, a deformation algorithm should have local influence and detail preservation to make the deformed mesh be similar to the original one. This is especially difficult when stretching or shrinking occurs on a part of the mesh.

Moreover, shape preservation is not the only demand for the deformation algorithm. A natural and smooth result is usually required for the editing. This is consistent with the aesthetics of the human being as well as the requirement of many practical applications such as character modeling in animation.

An inevitable problem then comes: What is a good balance between the rigidity and the fairness of the transformation? The answer to this problem is probably not unique. A satisfactory editing result depends not only on the algorithm adopted, but also the specific models and the desired effects the users need.

Instead of giving an absolutely clear answer, we aim to explore a method which could generate various potentially available results and leave the selection work, which is to some extent subjective, to the users. After all, all the results should satisfy the above-mentioned requirements for surface deformation.

In Section 2, some existing mesh deformation algorithms are reviewed, with a focus on how the related problems are addressed. Then in Section 3 we propose a new algorithm which considers the intrinsic geometric properties of the mesh surface from two different points of view. By combining the two geometric properties, a flexible deformation method that takes both the intrinsic shape and smoothness into consideration is developed. In Section 4, some examples are provided with some discussions. Finally Section 5 concludes the paper.

2. RELATED WORK

Surface editing of objects in 3D space has been studied for years. Early research for editing a surface was based on parametric surfaces [1]. In recent years, techniques for editing discrete form surfaces (especially triangular mesh) have drawn much attention from scientists and researchers and many algorithms have been developed. These algorithms dealt mainly with smooth surfaces [2, 3]. Similar to the work of editing parametric surfaces, they tried to keep the shape as smooth as possible while making modifications on it. The local shape and orientation are not considered.

Subsequent works tried to explore the intrinsic properties of the local surface. Among many kinds of geometric expressions of local region, mean curvature is mostly often considered. A pioneer work was proposed by Marc Alexa, which makes use of the discrete mean curvature to do mesh editing [4]. Its basic idea was to minimize the change of discrete mean curvature of each mesh vertex during deformation. The method is also called the Laplacian approach. An apparent drawback here is that this approach is only translation-invariant, but sensitive to rotation and scale transformations. After that, a lot of remedy methods were proposed to estimate the rotation and scale transformations [5, 6, 7, 8, 9]. However, while the Laplacian approach usually results in smooth deformed shapes, it is still not sufficient to preserve the local properties of the mesh.

To make the deformation more rigid, Olga Solkine *et*

al. proposed another kind of expression of the mesh local shape [10]. By defining a unit cell as a vertex and its direct neighbors, the method tried to keep each edge vector in a cell as fixed as possible under translation and rotation transformations. In this way, the deformation is much more rigid than the Laplacian deformation, but it is sensitive to scale transformation and sometimes the rigidity of the deformation may cause non-smooth appearance.

Radial basis functions were also employed by Mario Botsch *et al.* to implement mesh deformations [11]. The triharmonic basis function serves as the basic function which guarantees C^2 continuous boundary constraints. First a linear system is established based on the old and new positions of the handles and then the positions of other vertices are updated by solving the linear system. This method helps to get a smooth deformation, but some local shape information may get lost.

There is another method called multi-resolution mesh processing which divides the mesh into layers according to the level of geometric details [12, 13, 14]. The base level is the low-frequency component of the mesh and typically represented in Cartesian coordinates. The refinement levels above it are described locally so that the geometric details are mostly captured in a discrete set of translation and rotation-invariant coordinates. Using this representation, modeling operations can be performed on an appropriate user-specified level-of-detail. To use this method, one first needs to explicitly create a satisfactory multi-resolution hierarchy, which is actually not an easy task. If a mesh contains complex details, many levels may be required.

This paper focuses on developing a method that uses a single resolution representation and is capable of describing the local shape of the mesh. Specifically, we propose a new framework that combines the advantages of both the Laplacian and the rigid approaches and enables further adjustment to obtain the best visual result.

3. FLEXIBLE MESH DEFORMATION

3.1 Fundamentals

We first review the geometric meanings and mathematical expressions of the rigid and Laplacian deformations. The object surface could be considered as covered by many small cells. An important goal of mesh deformation is to preserve the shape of the cells.

For rigid transformation, the change of each edge vector between a vertex and its neighbors in a cell is minimized up to translation and rotation. Mathematically, let p_i be the position of vertex i and p_j the position of a neighbor, $e_{ij} = p_i - p_j$ be the edge vector between vertex i and vertex j . What the rigid transformation tries to minimize is:

$$E(e'_{ij}) = \sum_{j \in N(i)} \omega_{ij} ||e'_{ij} - R_i e_{ij}||^2 \quad (1)$$

where, $j \in N(i)$ means that vertex j shares a common edge with vertex i, e'_{ij} denotes the deformed vector, and R_i represents the rotation matrix of e_{ij}. Since the edges pointing

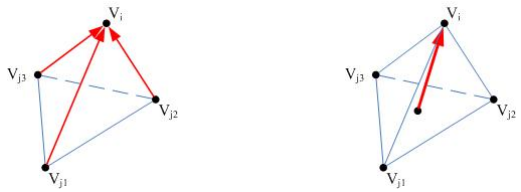


Fig. 1: A unit cell in mesh deformation. The rigid deformation tries to preserve the red edge vectors pointing to vertex v_i as shown in (a); The Laplacian deformation tries to preserve the red vector pointing to vertex v_i from the center of its neighbors as shown in (b).

to vertex i may have different lengthes, cotangent weight ω_{ij} for edge e_{ij} is defined according to the work by Meyer *et al.* [15].

By computing the partial derivative w.r.t the new position p'_i of vertex i and letting it be zero, we obtain the following linear equation for each vertex v_i:

$$\sum_{j \in N(i)} \omega_{ij}(p'_i - p'_j) = \sum_{j \in N(i)} \frac{\omega_{ij}}{2}(R_i + R_j)(p_i - p_j) \quad (2)$$

The rotation matrix R_i could be computed given the old and new values of the edge vectors. Detailed derivation of equation 1 and 2 could be found in [10]. From the definition it could be seen that rigid deformation strives to do a *hard* transform to the mesh. We say "hard" as the edge vectors of a vertex could completely determine the shape of a unit cell. As long as each vector is preserved, the shape of the cell will not change at all.

For Laplacian deformation, a vector called differential coordinate is defined as $\delta_i = \sum_{j \in N(i)} \omega_{ij}(p_i - p_j)$. The direction of δ_i approximates the local normal direction and the magnitude approximates a quantity proportional to the discrete local mean curvature. The Laplacian deformation minimizes the change of δ_i. However, as mentioned in Section 2, the differential coordinate is sensitive to rotation and scale transformations, so a matrix T_i is defined to compensate the change of δ_i and thus the following equation could be obtained for each vertex v_i:

$$\sum_{j \in N(i)} \omega_{ij}(p'_i - p'_j) = T_i \delta_i \quad (3)$$

It could be seen that Laplacian deformation is a relative *soft* deformation comparing to the rigid one, since the differential coordinate could not represent the shape of the cell exactly. The word "soft"has another meaning that the deformed shape tends to be smooth, since the change of mean curvature is minimized to prevent the occurrence of irregular shape within a local area.

3.2 A New Deformation Algorithm

For mesh deformation, we want not only the change of the shape to be minimized, but also a relatively smooth result without any irregular part caused by the deformation. However, it is difficult to achieve this goal by only using the rigid or Laplacian deformation. In the rigid deformation the definition of the local shape is quite accurate, so sometimes the result is so "rigid" that smooth result may not be available. Besides, a definite rigid transformation means that scale is not considered, as can be seen from equation 1, and thus some aliasing of the shape may occur under this kind of transformation. By contrast, the Laplacian deformation is based on an approximate expression of the discrete mean curvature. Thus the local shape is not so exactly preserved as the rigid transformation in this circumstance. However, a relatively smooth result which avoids the irregular appearance could be obtained using Laplacian deformation. Moreover, scale transformation is considered in the computation to make the deformation result more natural under stretching or shrinking.

To make use of the advantages of these two deformations and give the user a flexible choice of the desired result subject to the specific model, we propose a new mesh deformation method that combines these two deformations. Our aim is to control the result by adjusting a weight parameter λ ($0 \leq \lambda \leq 1$). When $\lambda = 1$, the deformation is the most rigid and when $\lambda = 0$ a smooth result while trying to preserve the original local geometric feature is achieved. Multiple in-between results are available for different intermediate λ values. To achieve this goal, we simply multiply both sides of equation 2 by λ and equation 3 by $(1-\lambda)$ respectively and then add the corresponding sides of the two new equations. It should be noted that the left hand sides of equation 2 and 3 have the same form: discrete Laplace-Beltrami operator. So the overall equation of our algorithm could be written as:

$$\begin{aligned} \sum_{j \in N(i)} \omega_{ij}(p'_i - p'_j) &= \lambda \sum_{j \in N(i)} \frac{\omega_{ij}}{2}(R_i + R_j)(p_i - p_j) \\ &+ (1-\lambda) T_i \delta_i \end{aligned} \tag{4}$$

where $\delta_i = \sum_{j \in N(i)} \omega_{ij}(p_i - p_j)$ is the differential coordinate defined in Laplacian deformation. It could easily be seen that when λ equals 0 or 1, the result is just the same as that of the Laplacian or rigid deformation.

The linear system (4) could further be written as the following compact form:

$$Lp' = r \tag{5}$$

where p' is a vector containing all the unknown new positions of the related vertices we want to solve, r is a vector which is composed of the right hand side values of equation 4 for each vertex, and L is the coefficient matrix that has the following form:

$$L_{ij} = \begin{cases} \sum_{j \in N(i)} \omega_{ij}, & i = j \\ -\omega_{ij}, & (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

In addition, to ensure that the handles interpolates the new positions specified by the users and the anchors stay fixed during the deformation process, the following constraints for the handle and anchor vertices need to be introduced:

$$Ip'_j = c \tag{6}$$

where I is the identity matrix, p'_j is a vector consisting of all the unknown new positions of the handle and anchor vertices, and c denotes their designated new positions after deformation.

In previous work, the matrices in equation 5 and 6 are combined into one matrix and a large value of weight is given to latter one if more accurate interpolation of the handle vertices is required. However, this is insufficient if exact interpolation is needed. So in our system, we treat equation 6 as a strict constraint and convert the problem into a constrained least squares problem:

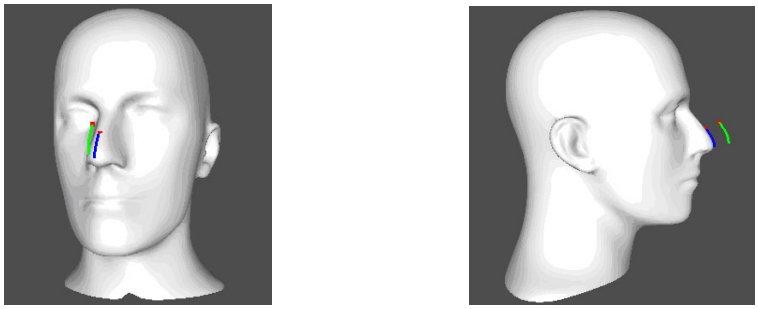
$$\begin{aligned} &\min \ (Lp' - r)^2 \\ &\text{s.t.} \ Ip'_j = c \end{aligned} \tag{7}$$

In this way, accurate interpolation of the handle vertices could be achieved such that a curve drawn by the user could be exactly the new positions of the handle vertices after deformation. This can be seen from the results in the next section.

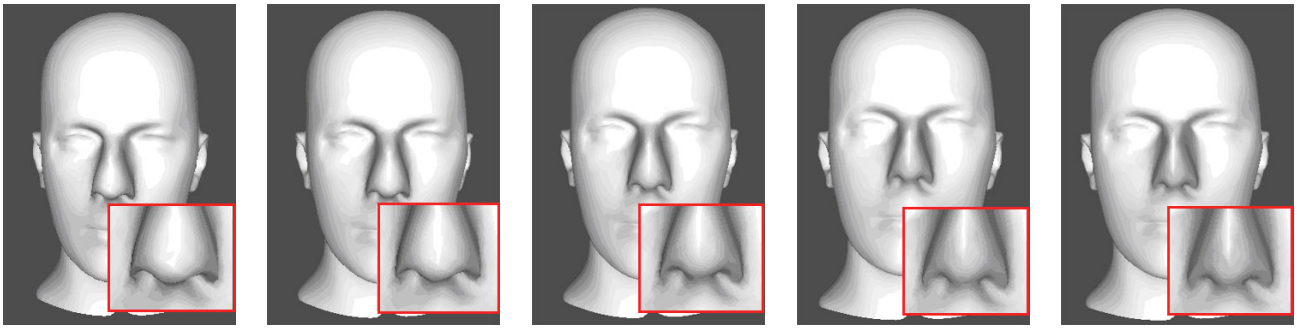
In summary, our algorithm can be outlined as follows. First the weight for each edge and the system matrix of the left hand side of equation 5 are precomputed. These items stay fixed during the whole calculation process. Then an initial deformation result is estimated using the naive Laplacian deformation which does not consider rotation and scale transformations. Using the initial result, we could compute the rotation matrix R_i and the transformation matrix T_i. Next the new positions of the vertices are updated through equation 4. Finally some iterations are performed for recomputing R_i and T_i and then a new right hand side of equation 4. This process continues till a satisfactory result is obtained. It should be pointed out that in this algorithm the choice of value of coefficient λ depends on the specific model and the user desired deformation effect.

3.3 Computation of Transformation Matrix T_i

Note that in the algorithm presented in the preceding subsection, we need to compute the transformation matrix T_i. There have been some existing methods of computing T_i.



(a) Two views of the original model



(b) $\lambda = 0$ (c) $\lambda = 0.2$ (d) $\lambda = 0.5$ (e) $\lambda = 0.8$ (f) $\lambda = 1$

Fig. 2: Deformation results of the *mannequin* model under different λ values. With the increase of the value, the shape of the nose becomes thinner.

[5] calculates T_i by estimating the local rotations of each vertex implicitly and [6] computes T_i explicitly by using a matrix which contains the new positions of the vertices to evaluate the geometric change of a vertex and its neighbors and then equation 3 contains only one unknown vector–the new positions of the vertices.

However, both these representations of T_i are not accurate enough to evaluate the change of the differential coordinate δ_i under rotation and scale transformations. We decompose T_i into a scale transformation S_i and a rotation transformation R_i such that $T_i = S_i R_i$. Similar to the method in the rigid deformation [10], we evaluate the rotation matrix R_i according to the change of the edge vectors of a cell. The task left then is to compute the appropriate matrix S_i.

Assume that in equation 1 the old and new edge vectors and the rotation matrix have been known, then what we want to find is a scale matrix S_i which minimizes the following expression:

$$E(S_i) = \sum_{j \in N(i)} \omega_{ij} ||e'_{ij} - S_i R_i e_{ij}||^2 \qquad (8)$$

where S_i is in the form of $\begin{pmatrix} S_{ix} & 0 & 0 \\ 0 & S_{iy} & 0 \\ 0 & 0 & S_{iz} \end{pmatrix}$. S_{ix}, S_{iy} and S_{iz} are the unknown scaling coefficients corresponding to the x, y and z coordinate directions. By computing the partial derivatives w.r.t. S_{ix} , S_{iy} and S_{iz} respectively and making them equal zero, the unknowns in the scale matrix could be calculated.

It can be seen that as long as we get the old and new edge vectors and the corresponding weights, we could compute T_i. This method is more precise for estimating the change of the differential coordinates than the previous work. It is also easy to implement, thus providing a better choice for our deformation algorithm.

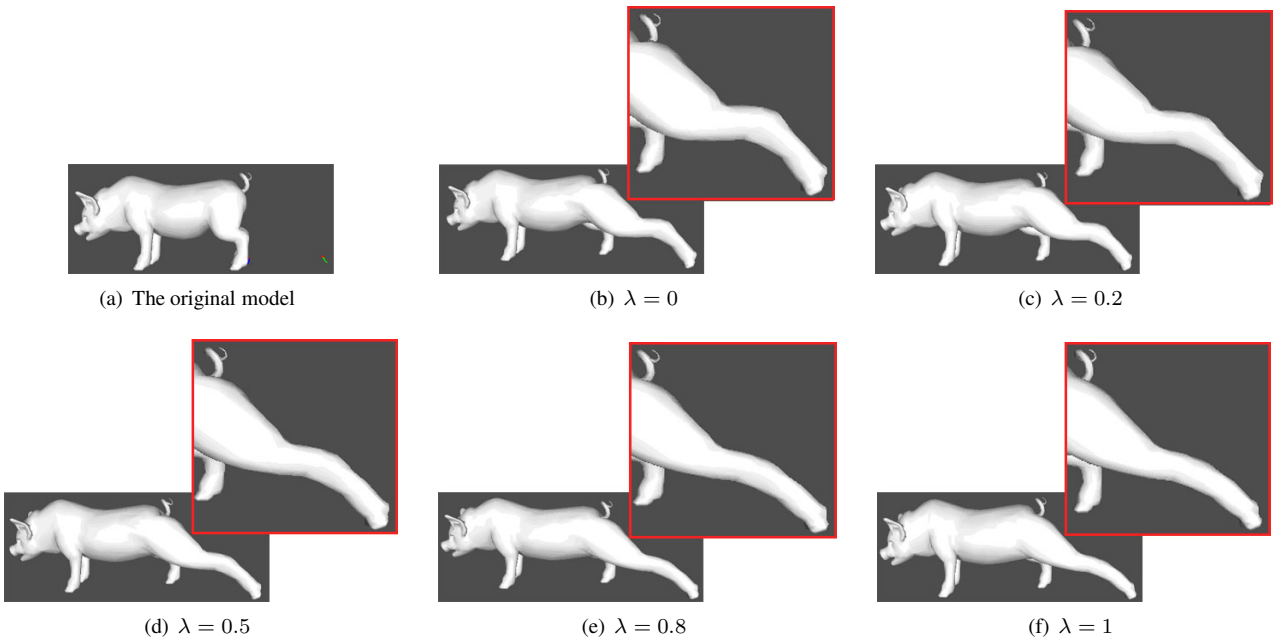
4. RESULTS AND DISCUSSION

We have implemented our deformation algorithm using C++ on a Pentium 4 3.6G computer with 1GB RAM. For solving the linear equations, we used the LAPACK library [16].We demonstrate that our algorithm is effective and flexible for users to adjust the visual deformation results.

In figures 2- 4, the blue lines indicate the handle vertices on the mesh and the green lines are the designated positions of the handles. The red dots indicate the starting point of them. It has been mentioned in Section 3.2 that we solved a constrained least squares problem such that the positions of the handle vertices in the deformed mesh exactly interpolate the green lines we specified.

Figure 2 shows the deformation results of the model *mannequin*. We stretch the nose tip to make it a Roman nose. It could be seen that the deformed nose becomes sharper with the increase of the λ value. This is because the Laplacian approach tends to make the deformed shape smooth and fat while the rigid deformation tries to preserve the shape without considering the smoothness. In practice, the user can choose his/her preferred shape among all these deformation results.

In figure 3 we pull the pig's left hind leg to the position marked by the green line. It is observed that when the value of λ is large, the deformed shape is aliasing. This is due to



(a) The original model (b) $\lambda = 0$ (c) $\lambda = 0.2$

(d) $\lambda = 0.5$ (e) $\lambda = 0.8$ (f) $\lambda = 1$

Fig. 3: Stretching the left hind leg of the *pig* model. Different results corresponding to different λ values are shown from (b) to (f). For larger values, the shape of left hind leg looses its original characteristics

the fact that the rigid deformation is sensitive to scale transformation. When stretching occurs, the rigid deformation is unable to keep the shape as it was. Whereas for the result of $\lambda = 0$, the shape is smooth and round. Obviously this is also against the actual physical phenomenon that when an object is stretched, it tends to be longer and thinner instead of keeping the shape as it was. From this point of view, $\lambda = 0.2$ or $\lambda = 0.5$ might be a good choice.

We rotate and extend the upper mouth of the small dragon in figure 4. From $\lambda = 0$ to $\lambda = 1$, the upper mouth gradually becomes thinner. Combining the considerations of keeping the shape similar to the original while not losing smoothness, $\lambda = 0.5$ may be a good choice.

In our implementations we performed only two iterations for calculating the rotation and scale matrices and updating the new positions of the related vertices. For deformations that are not quite large, this is enough to gain a good result. In the experiment, our system runs interactively for regions of interest (ROI) with up to more than two thousand vertices. This provides the users intuitive visual feedbacks and meets the requirement of mesh editing techniques. However, we observed that for larger rotation and scale, more iterations were needed to calculate a satisfactory result. Moreover, if a larger size of ROI is involved, the computation time may be longer. In this case, the performance of the algorithm could be improved by some optimization techniques, such as using the multi-resolution algorithms.

We also noted that the decision of the best editing result for different models does not have a unique objective rule. It depends not only on the shape of the model we wish to deform, but also the aethestic standard and specific demands of the users. Taking the small dragon model for example, if a realistic effect is desired for the editing, then $\lambda = 1$ seems a good choice; if the user wants a cartoonish result, then $\lambda = 0$ gives the desired result. The availability of the weight parameter does provide the user flexibility to choose a desired shape from multiple possible results.

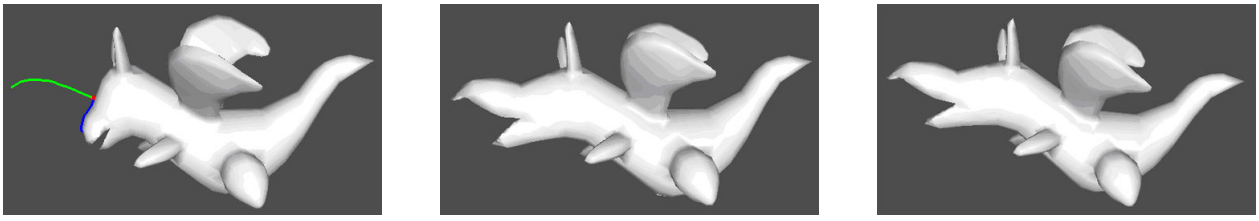
5. CONCLUSION

Mesh deformation has been a difficult problem in geometric processing since it requires to produce a shape that is not only similar to the original one but also with natural and smooth appearance. We have presented a flexible deformation scheme to enable the adjustment between the rigidity and smoothness of the deformation result. Several examples have demonstrated that our algorithm is able to provide flexibility for users to choose the desired deformation results. This algorithm could be further extended for some other applications such as generating keyframes for computer-assisted animation making.

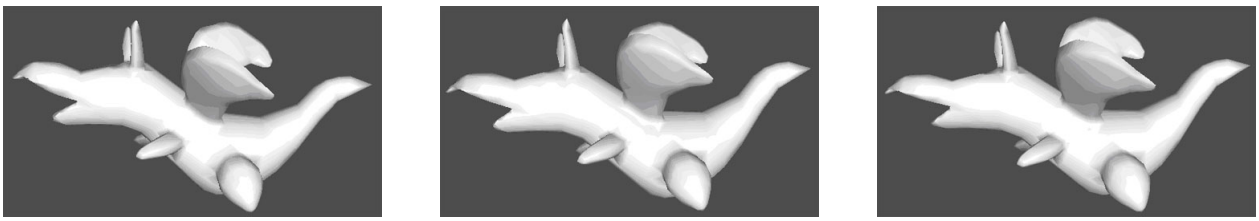
However, there are still some issues that need to be explored in further research on shape deformation. The method of computing the rotation and scale matrices is time-consuming when the number of vertices becomes larger. It is important to find a faster yet precise way to compute it. The Laplacian deformation ignores the distribution of triangles with different sizes. This may cause some unsatisfactory results at the area where the mesh is dense, so appropriate weights should be added according to the density of the triangles.

Acknowledgments

This work is supported by the ARC 9/09 Grant (MOE2008-T2-1-075) of Singapore.



(a) The original model (b) $\lambda = 0$ (c) $\lambda = 0.2$



(d) $\lambda = 0.5$ (e) $\lambda = 0.8$ (f) $\lambda = 1$

Fig. 4: Editing the small dragon model. The upper mouth is pulled and extended. Different deformation results of different λ values are shown from (b) to (f).

6. REFERENCES

[1] Gerald Farin. "Curves and surfaces for computer aided geometric design (3rd ed.): a practical guide," Academic Press Professional,1993.

[2] William Welch and Andrew Witkin. "Free-form shape design using triangulated surfaces," SIGGRAPH'94, pp. 247–256, 1994.

[3] Gabriel Taubin. "A signal processing approach to fair surface design," SIGGRAPH'95, pp. 351–358, 1995.

[4] Marc Alexa. "Differential coordinates for local mesh morphing and deformation," The Visual Computer'03, Vol. 19, No. 2-3, pp. 105–114, 2003.

[5] Yaron Lipman., Olga Sorkine., Daniel Cohen-Or., David Levin., Christian Rössl, and Hans-Peter Seidel. "Differential Coordinates for Interactive Mesh Editing," IEEE SMI'04, pp. 181–190, 2004.

[6] Olga Sorkine., Yaron Lipman., Daniel Cohen-Or., Marc Alexa., Christian Rössl, and Hans-Peter Seidel. "Laplacian Surface Editing," Eurographics SGP'04, pp. 179–188, 2004.

[7] Yaron Lipman., Olga Sorkine., Daniel Cohen-Or, and David Levin. "Linear Rotation-Invariant Coordinates for Meshes," SIGGRAPH'05, pp. 479–487, 2005.

[8] Andrew Nealen., Olga Sorkine., Marc Alexa, and Daniel Cohen-Or. "A Sketch-Based Interface for Detail-Preserving Mesh Editing," SIGGRAPH'05, pp. 1142–1147, 2005.

[9] Yu Yizhou., Zhou Kun., Xu Dong., Shi Xiaohan., Bao Hujun., Guo Baining, and Shum Heung-Yeung. "Mesh editing with poisson-based gradient field manipulation," SIGGRAPH'04, pp. 644–651, 2004.

[10] Olga Sorkine and Marc Alexa. "As-Rigid-As-Possible Surface Modeling," Eurographics SGP'07, pp. 109–116, 2007.

[11] Mario Botsch and Leif Kobbelt. "Real-Time Shape Editing using Radial Basis Functions," Eurographics '05, pp. 611–621, 2005.

[12] Denis Zorin., Peter Schröder, and Wim Sweldens. "Interactive multiresolution mesh editing," SIGGRAPH'97, pp. 259–268, 1997.

[13] Igor Guskov., Wim Sweldens, and Peter Schröder. "Multiresolution signal processing for meshes," SIGGRAPH'99, pp. 325–334, 1999.

[14] Mario Botsch and Leif Kobbelt. "An intuitive framework for real-time freeform modeling," SIGGRAPH'04, pp. 630–634, 2004.

[15] Mark Meyer., Mathieu Desbrun., Peter Schröder, and Alan H. Barr, et al. "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds," Visulization and Mathematics *III*'03,pp. 35–57, 2003.

[16] Anderson E., Bai Z., Bischof C, and Blackford S., et al. "LAPACK Users' Guide," Society for Industrial and Applied Mathematics'99,1999.