Edge-based Laplacian Mesh Deformation

Kai Wang [*] Jianmin Zheng [†] Hock-Soon Seah [‡]

School of Computer Engineering
Nanyang Technological University, Singapore

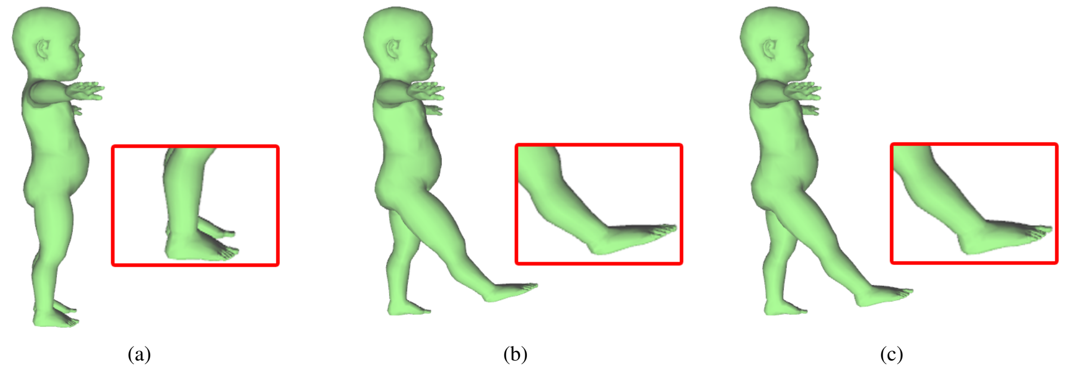

Figure 1: *Deformation of a baby model: (a) The initial model; (b) The deformed result generated by the vertex-based Laplacian deformation algorithm; (c) The deformed result generated by our edge-based Laplacian deformation algorithm.*

Abstract

Mesh deformation is an important model editing process in geometric modeling and computer animation. It provides a convenient way to modify a given mesh to meet various design requirements in many applications. Usually, a deformation algorithm is expected to be able to present a natural and smooth deformation result while preserving the intrinsic geometry details of the mesh as much as possible. Most of the existing mesh deformation methods establish their formulations directly in the vertex-based domain. This sometimes has a drawback that a satisfactory deformation may not be achieved due to the insufficient number of points that are used in the deformation process for the shape details. In this paper, we propose to use edges of the primal mesh as nodes to construct a new graph representation for the mesh and then to perform Laplacian deformation in the edge-based domain, presenting a new mesh deformation method, which we call the edge-based deformation algorithm. Compared with those vertex-based Laplacian deformation algorithms, the edge-based deformation algorithm uses roughly thrice the number of nodes in the deformation process and thus well preserves the local geometry details of the mesh. The experimental examples demonstrate that our edge-based algorithm in general produces better deformation results than the vertex-based Laplacian algorithms.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations;

Keywords: Computer graphics, geometric modeling, mesh deformation

[*]e-mail: wang0452@ntu.edu.sg
[†]e-mail: asjmzheng@ntu.edu.sg
[‡]e-mail: ashsseah@ntu.edu.sg

1 Introduction

Surface deformation is an effective and convenient method for mesh editing in geometric modeling. It allows direct manipulation of the mesh surface and gives direct visual feedback of the operations. It has a wide range of applications in industrial and artistic design. Due to its importance and popularity, many research have been carried out and various algorithms have been proposed. Though having their respective advantages, those algorithms still suffer various problems, owing to some considerations in surface-based mesh deformation as listed below.

First, since a polygonal mesh can be considered as the discretization of a smooth (piecewise smooth) surface, its shape is only an approximation of the smooth one and this approximation depends on the sampling rate – in general the higher sampling rate is adopted, the better approximation effect is achieved. When a deformation is performed on a mesh, the number of points sampled from the shape for the deformation calculation affects the deformation result. However it is not trivial to have more points be involved in the deformation within a given deformation region.

Second, the local geometric shape of the deformed mesh should be similar to that of the original mesh. For this purpose, local shape features, which are represented by the local geometric quantities such as curvature and normal and calculated at the vertices of the mesh, should be well preserved. Since translation and rotation transformations will not change the geometric shape, in surface deformation the preservation of the 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. In general, a deformation algorithm should have local control 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. Besides, a natural and smooth result is often required for the editing, which conforms to the aesthetics of the human beings as well as the requirement of many practical applications such as character modeling in animation.

Third, in surface deformation typically the user is required to directly manipulate the mesh surface to adjust the shape. Thus quick and intuitive visual feedback is needed. This makes it necessary for the deformation algorithms to have computation completed in a very short time period. To achieve this requirement, most of the algorithms try to formulate their problems as a linear one and solve it using various advanced linear solvers.

For most of the existing mesh deformation algorithms, local shape preservation of the mesh is achieved by minimizing the change of the shape of a small neighborhood area at each vertex. However, we find that sometimes these vertex-based methods cannot preserve the local shape details well. This is partially because the number of vertices is not large enough for defining the shape of the mesh. In this paper, we propose a new approach for mesh deformation, which is based on the edge-based representation. This approach takes edges as nodes and constructs a new graph representation for the mesh. Since the number of edges on a mesh is roughly thrice that of vertices, more points will be involved in the evaluation of the shape of the mesh if we formulate the deformation problem in the edge domain. By applying the popular Laplacian deformation method to the edge-based representation, our algorithm could obtain better deformation results than the existing vertex-based deformation algorithms.

The rest of the paper is organized as follows: Section 2 reviews some existing mesh deformation algorithms, with an emphasis on how the relevant problems are addressed. In Section 3 we present the edge-based Laplacian deformation algorithm. Experimental results are provided with discussions in Section 4. Finally Section 5 concludes the paper.

2 Related Work

The deformation of objects in 3D space has been studied for years. The surface deformation methods for parametric or subdivision surfaces have already been developed. But these algorithms are difficult to be applied directly to triangular meshes. With the generalization of digital geometry media, mesh deformation has become a popular research topic in the field of geometric modeling in recent years.

Existing methods can be divided into two categories: the space-based deformation [Gain and Bechmann 2008] and the surface-based deformation [Botsch and Sorkine 2008]. The space-based deformation methods indirectly reshape an object by warping its surrounding space, with results that are similar to modeling a highly malleable substance. They have the virtue of being computationally efficient and applicable to a variety of object representations. However in these methods, when large scale deformation happens, the geometric details on a mesh could not be well preserved. Besides, it is not suitable for applications where direct manipulations on the mesh surfaces are needed.

Contrary to spatial deformation, surface-based methods carry out the mesh deformation by allowing direct manipulation of the mesh surfaces. The most popular approach is the Laplacian coordinates, which encodes the mesh in an intrinsic manner. For each vertex of the mesh, a so-called Laplacian vector is calculated. This vector starts from the center of the neighbors to the vertex itself. The magnitude of the vector approximates the mean curvature at the vertex and its direction represents the normal direction. Therefore the Laplacian vector contains the local geometric shape information at each vertex of the mesh. Mesh deformation is implemented by minimizing the changes of the Laplacian vectors [Alexa 2001].

However, since the Laplacian vector is only translation-invariant and will be changed under rotation and scale transformations, the local transformation of the vector during deformation needs to be estimated. Remedial methods have been proposed to estimate the rotation and scale transformations. These methods can be further classified into explicit and implicit types.

Explicit methods estimate the local transformation without considering the deformed surface. It propagates the transformations at the handles to all the unconstrained vertices, weighted by geodesic distances [Yu et al. 2004; Zhou et al. 2005] or harmonic fields [Zayer et al. 2005]. These methods produce good results for large rotations. But when the handle vertices are only translated, there would be no change of orientation to be propagated.

Implicit methods evaluate the transformation according to the deformed surface. This is in fact a *chicken-and-egg* problem: on one hand, the deformed mesh is reconstructed from the transformed Laplacian vectors; on the other hand, the transformation is derived given the initial and deformed surface.

Lipman et al. [Lipman et al. 2004] calculated the local rotation between the original mesh and an initial deformation result by a locally estimated coordinates. It works well for relatively smooth surfaces with no largely protruding features.

Sorkine et al. [Sorkine et al. 2004] tried to represent the local transformation at each vertex as the unknown position of vertices and their original positions. Then the deformed mesh could be calculated by solving just one linear system. However, the linearization of the local transformation makes it difficult to handle large rotations.

To overcome this problem, Fu et al. [Fu et al. 2007] proposed to first solve a local affine transformation T_i associated to each vertex and then derive a similarity transformation from the solved affine-transformations of the vertex and its neighbors to transform the Laplacian coordinates for the subsequent mesh reconstruction. This approach allows large rotations, but it requires tweaking the relative weighting of local transformation smoothness terms. Moreover, only isotropic scale is considered while anisotropic scale often obtains better results.

All the above mentioned Laplacian deformation methods are based on the vertex-based mesh representation, in which the Laplacian vector is calculated at each vertex. However, since the number of vertices of a mesh is limited, the "constraints"for defining the local shape feature are sometimes insufficient. In that case, satisfactory deformation results may not be obtained.

This paper presents a new surface-based mesh deformation method. Specifically, we extend the Laplacian mesh deformation algorithm from the vertex-based to the edge-based domain, to obtain better deformation results by preserving the local details and keeping global smoothness.

3 Edge-based Laplacian Mesh Deformation

3.1 Fundamentals

We begin with the basic vertex-based Laplacian coordinates. Let $M = (V, E)$ be a triangular mesh with n vertices. $V = \{v_i | v_i \in R^3, i = 1, ..., n\}$ denotes the set of vertices and $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, i \neq j\}$ denotes the set of edges. Each vertex $v_i \in V$ is conventionally represented using absolute Cartesian coordinates, denoted by $v_i = (x_i, y_i, z_i)$. Mathematically, the Laplacian coordinate δ_i at v_i is defined as:

$$\delta_i = D(v_i) = \sum_{j \in N(i)} \omega_{ij}(v_i - v_j), \tag{1}$$

where $N(i) = \{j|e_{ij} \in E\}$ is the index set of adjacent vertices in the *neighboring ring* of vertex v_i and ω_{ij} is the weight of the edge e_{ij}. The *neighboring ring* is also called the 1-ring neighbors.

A simple scheme for computing the weight ω_{ij} in Equation 1 is to use the uniform weight, where $\omega_{ij} = 1/|N(i)|$. This scheme works well for the mesh with relatively regular topology and geometry. However for irregular meshes, the calculation using uniform weights could not exactly represent the normal and curvature information at each vertex. So a more general scheme is to use the cotangent weight:

$$\omega_{ij} = \frac{1}{\Omega_i}(\cot \alpha_{ij} + \cot \beta_{ij}), \tag{2}$$

where Ω_i is the area of the Voronoi cell associated with v_i, and α_{ij}, β_{ij} are the two angles opposite to e_{ij}. An example of the Laplacian vector at a vertex can be seen in Figure 2.

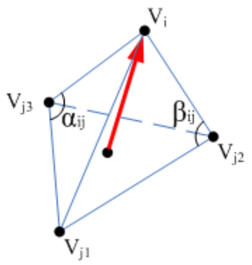

Figure 2: *The Laplacian vector at vertex v_i is the red vector pointing to v_i from the weighted center of its neighbors. α_{ij} and β_{ij} are the two angles used for calculating the weight for the edge between v_i and v_{j1}.*

Using Equation 1, we could represent the mesh geometry using a set of Laplacian coordinates $\Delta = \{\delta_i\}$. For a given mesh M, the transformation between V and Δ could be described through the following equation:

$$LV = \Delta, \tag{3}$$

where L is an $n \times n$ coefficient matrix (n denotes the number of vertices) that has the following form:

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

It is noted that L has rank $n-1$, which means V can be recovered from Δ by fixing one vertex and solving a linear system.

3.2 Laplacian Mesh Deformation

The approach of performing a mesh deformation task using Laplacian coordinates designates the new absolute positions $\{c_i\}$ of several vertices (see [Lipman et al. 2004]), i.e.:

$$v'_i = c_i, \qquad i \in \{m, ..., n\}, m < n \tag{4}$$

and solves for the remaining vertices $\{v'_i\}, i \in \{1, ..., m-1\}$ by fitting the new Laplacian coordinates of the deformed mesh to the given Laplacian coordinates of the original mesh. The vertices whose positions are to be designated usually include the *handle* vertices, which the users want to move to designated new 3D locations and the *static* vertices, which are expected to stay fixed during the deformation process. The other vertices whose positions need to be solved are usually called the *Region of Interest* (ROI) vertices.

It has been observed that the solution behaves better if the positional constraints $\{c_i\}$ are satisfied in a least square sense rather than exactly [Lipman et al. 2004; Sorkine et al. 2004]. This results in the following object function

$$E(V') = \sum_{i=1}^{n} ||\delta_i - D(v'_i)||^2 + \sum_{i=m}^{n} ||v'_i - c_i||^2, \tag{5}$$

which should be minimized. The new position v'_i of each vertex after deformation could then be obtained by solving a sparse linear system.

The rationale of this method is to preserve the local details of the shape through minimizing the change of the Laplacian vectors during the deformation process. However, it has been mentioned that the Laplacian vector is not invariant under rotation and scale transformations. So if a deformation that contains rotation or scale happens on a given mesh, the method of Equation 5 will try to preserve the orientation and magnitude of the Laplacian vectors w.r.t. the global coordinate system, whereas in reality these vectors should be rotated or scaled. So the transformations of the Laplacian vectors during the deformation process should be considered. Assume the transformation matrix for a Laplacian vector δ_i be T_i, then the object function which has to be minimized for solving the deformation problem becomes:

$$E(V') = \sum_{i=1}^{n} ||T_i\delta_i - D(v'_i)||^2 + \sum_{i=m}^{n} ||v'_i - c_i||^2. \tag{6}$$

Besides the local transformation problem, there is another important issue which is often ignored in existing algorithms: the number of *points* for defining the shape of the mesh surface. As has been introduced, the mesh is a discretization of the continuous surface, and the approximation effect depends on the sampling rate. In existing vertex-based Laplacian deformation methods, the number of *points* is just the number of vertices. For a mesh with limited number of vertices, these approaches sometimes cannot preserve the shape of the mesh surface well, since not enough points are involved in the evaluation of the local shape features.

To provide a method of choosing more points on the mesh surface reasonably for the evaluation of the local shape properties, we propose the edge-based Laplacian deformation algorithm, which will be introduced in the next section.

3.3 Edge-based Mesh Representation

The edge-based mesh representation takes edges as basic primitives. The local shape feature of the mesh surface is evaluated at the midpoint $m_{ij} = (v_i + v_j)/2$ of each edge e_{ij}, with neighborhood consisting of all the midpoints of the other edges that share a same vertex with e_{ij}. Such an example is shown in Figure 3.

Since the number of edges on a mesh is approximately three times that of vertices, more points are involved in the evaluation of the local shape feature of the mesh surface. Intuitively speaking, there will be more “constraints”for preserving the shape of the mesh for

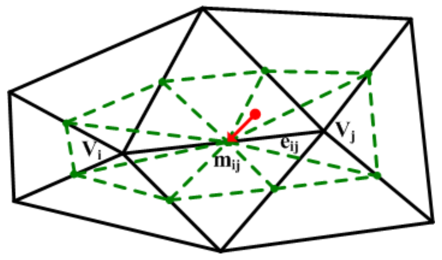

Figure 3: *The edge-based representation. The black triangles are the faces of the original mesh. The green dashed lines connect the midpoint m_{ij} of edge e_{ij} and the midpoints of the edges that contain vertex v_i or v_j. The red arrow represents the Laplacian vector at m_{ij} in the edge-based domain.*

a deformation task. For a mesh with n vertices, the edge number should be approximately $3n$. For simplicity, let us assume that the whole mesh is involved in a deformation operation. Then the shape of the mesh is defined by $3n$ points on the mesh surface in the edge-based method, whereas only n points are involved in the vertex-based approach. This rule also holds for the common case where a part of the mesh is involved in the deformation. With more points involved in the evaluation of the local geometric properties, the shape of the mesh is expected to be preserved better in our approach.

It should be noted that we do not introduce any extra vertices or change the mesh topology for the evaluation of the shape of mesh surface and only make use of the existing primitives of the mesh. From this perspective, the edge-based representation is a natural choice for better preservation of the mesh shape during deformation.

Specifically, to preserve the local shape feature and obtain a smooth result for mesh deformation, we apply the vertex-based Laplacian deformation algorithm to the edge-based representation. The Laplacian vector, which represents the local shape feature, is calculated for each edge of the mesh. Mesh deformation is done with the aid of the Laplacian vector computed at the midpoint of each edge of the original mesh. Similar to the vertex-based method, we could still establish a linear system, with the new positions of the vertices as unknowns. In the new system, there will be more rows in the coefficient matrix L, due to the larger number of edges. Besides, L will become denser, since comparing to the vertex-based method where only 1-ring neighborhood is included for computing the Laplacian vector at each vertex, more vertices are involved in the calculation of the Laplacian vector of each edge, which means that each row will have more non-zero elements. But generally speaking, L is still a sparse matrix in which the number of zero elements is far more than that of the non-zero elements. This means that the factorization of L, which is the most time-consuming step in solving the system would not take much time with advanced linear system solvers. The calculation of the other parameters is the same as that in the vertex-based Laplacian approach.

An important issue of the edge-based Laplacian deformation algorithm is the estimation of the local transformation matrix T_i. Next we will introduce our method of computing T_i in detail.

3.4 Computation of Local Transformation Matrix T_i

As has been introduced in Section 2, there have been many attempts on estimating the local transformation of the Laplacian vector, including the explicit and implicit methods. However, we still find that all these methods are not precise enough to evaluate the change of the Laplacian vector δ_i, since for the transformation none of them considered anisotropic scale, which is actually more general than isotropic scale. So we propose a method of calculating T_i by estimating the transformation of the center point which the Laplacian vector and its neighbor point to. Especially, we consider anisotropic scale in T_i, instead of isotropic scale.

Suppose the Laplacian vector at *vertex* v_i (the *vertex* here could be either the mesh vertex in the vertex-based representation, or the midpoint of the edge in our edge-based method) of the original mesh be δ_i, and the corresponding vector of the deformed mesh be δ_i'. The transformation matrix T_i is applied on δ_i to make it become δ_i'. In our algorithm, T_i is estimated through the change of the shape of a *cell*, represented by the edge vectors connecting v_i. We use the terminology *cell* to describe the shape composed of v_i and its 1-ring neighbors henceforth. Examples of a cell in the vertex-based and edge-based representations are shown as the blue lines in Figure 2, and the green dashed lines in Figure 3 respectively.

The object function we aim to minimize for computing T_i is:

$$E(T_i) = \sum_{j \in N(i)} \omega_{ij} ||e_{ij}' - T_i e_{ij}||^2, \tag{7}$$

where e_{ij} and e_{ij}' denote the original and deformed edge vectors. ω_{ij} is the corresponding weight for e_{ij}.

Since the order of various kinds of transformations is not unique, all we can infer is that T_i should be the combination of a series of rotation and scale matrices. It should be noted that for the measurement of the transformation of a cell, anisotropic scale is obviously more accurate and represents a more general case. So we only consider the anisotropic scale in T_i.

An inevitable problem then comes where the simplification of T_i becomes difficult. As we know that the multiplication of rotation and isotropic scale matrices can be finally simplified as a rotation matrix multiplying a scale factor. But this rule does not fit for the anisotropic scale case, and it is difficult to separate the rotation and anisotropic scale matrices from T_i.

But if we consider the problem from the theory of singular value decomposition for an arbitrary real value matrix, we could find that T_i can be represented by the multiplication of a rotation matrix, a scale matrix and another rotation matrix. So the object function for computing T_i becomes:

$$E(R_{2i}, S_i, R_{1i}) = \sum_{j \in N(i)} \omega_{ij} ||e_{ij}' - R_{2i} S_i R_{1i} e_{ij}||^2, \tag{8}$$

in which R_{2i} and R_{1i} are the two rotation matrices and S_i represents the scale matrix 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 of Cartesian coordinates.

We then use an iterative method to solve the three unknown matrices separately. We first replace T_i with R_{1i} to get an optimal rotation matrix, using the method proposed in [Sorkine and Alexa 2007]. After obtaining R_{1i}, we proceed to solve for S_i and R_{2i} one by one: first we treat R_{2i} as an identity matrix and S_i as unknowns. By putting the calculated R_{1i} into Equation 8, computing the partial derivatives w.r.t. S_{ix}, S_{iy} and S_{iz} respectively and making them equal to zero, the unknowns in S_i could be calculated. Then we put the calculated R_{1i} and S_i in and solve R_{2i}, using the same way as we calculate R_{1i}.

However during our experiments, we find that applying R_{2i} to the Laplacian vector has little effect on the final result and makes the computation time longer. This is because the computed S_i and R_{2i} are suboptimal (it is difficult to compute an optimal T_i directly when S_i is an anisotropic scale matrix) and their effects on the final result will decrease. So we just approximate T_i by the multiplication of S_i and R_{1i} and ignore R_{2i}.

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 Laplacian vectors than previous works and could reach convergence within a few iterations. 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 an Intel Xeon 2.27 G computer with 2GB RAM. For solving the linear equations, the LAPACK library [Anderson et al. 1992] is used. Some deformation results of the edge-based and vertex-based algorithms are shown in Figures 1, 4 and 5. The method of calculating the local transformation matrix T_i introduced in Section 3.4 is used in both edge-based and vertex-based algorithms.

Figure 1 shows the result of lifting a baby's leg. It could be seen that with the vertex-based method, both the shapes of the foot and leg are distorted, whereas with the edge-based method the local shape features are well preserved and the result is globally smooth.

In Figure 4 we pull the tail of a dinosaur. It is observed that the tail becomes "fatter"in the vertex-based method, which means that the local shape details of the tail are lost to some extent. Since the tail of the model contains much more details, the advantage of our algorithm is clearly demonstrated.

We pull the nose up in Figure 5. It turns to be a smoother nose in the vertex-based method, while in the edge-based method the rigidity of the nose is retained after deformation. However, in this example it is not quite easy to tell which is a better result, since it depends on the specific requirements of the user. Although the result of the vertex-based method is not that similar to the original shape as that of the edge-based method, it is still an alternative choice.

In our experiments we performed three to four iterations for calculating the rotation and scale matrices and updating the new positions of the related vertices. This has proved to be enough to gain a good result. Our system runs interactively for region of interest (ROI) with up to more than two thousand vertices (i.e. more than about six thousand edges). This proves that the increasing of the size and non-zero elements number of the coefficient matrix does not affect the speed too much. However if more vertices are involved, the computation time would be longer. In that case, the performance of our algorithm could be improved by some optimization techniques, such as parallel computing and so on.

It could be seen that for the applications where the preservation of the local shape is highly desired, our edge-based algorithm could produce obviously better result. This is due to the advantages of the edge-based representation we introduced. For the case that the meshes are with simple shapes and less details, or the requirement on the running speed of the program is quite strict, the vertex-based Laplacian deformation algorithm is also useful, since it could produce similar result while saving much computation time.

5 Conclusion

Mesh deformation is a difficult problem in geometric modeling due to the nature of the mesh representation and the difficulties of preserving the local geometric properties. In this paper we proposed a new surface representation which takes edges as basic primitives. We compute the Laplacian vector at the midpoint of each edge and by estimating the transformations of the Laplacian vectors and making use of the transformed vectors, improved deformation results could be obtained. Specifically, our algorithm gives an attempt to solve the following problems in mesh deformation:

1. How to reasonably include as many points as possible in the evaluation of the shape of a given mesh.
2. How to preserve the shape features at these points during deformation.
3. How to make the computation time as short as possible when solving the above two problems.

Experimental results have demonstrated the advantage of our algorithm over the existing vertex-based methods.

There are some issues that need to be explored in future. The increased computation cost brought by solving a larger linear system in the edge-based Laplacian method is expected to be further reduced for the interactive editing of larger meshes. For the computation of the local transformation matrix, the calculation of the optimal rotation and anisotropic matrices still remains an open problem. In addition, there may exist other methods for the more accurate calculation of the local geometric properties of the mesh surface.

Acknowledgements

This work is supported by the National Research Foundation grant, which is administered by the Media Development Authority Interactive Digital Media Programme Office, MDA (IDMPO), and by the ARC 9/09 Grant (MOE2008-T2-1-075) of Singapore.

References

ALEXA, M. 2001. Local control for mesh morphing. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications 2001*, 2–9.

ANDERSON, E., BAI, Z., BISCHOF, C., DEMMEL, J., DONGARRA, J., CROZ, J. D., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., OSTROUCHOV, S., AND SORENSEN, D. 1992. *LAPACK's user's guide*. Society for Industrial and Applied Mathematics.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 213–230.

FU, H., AU, O., AND TAI, C. 2007. Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Computer Graphics Forum 26*, 1, 34–45.

GAIN, J., AND BECHMANN, D. 2008. A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics 27*, 4, 1–21.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H. 2004. Differential coordinates for interactive mesh editing. In *SMI '04: Proceedings of the International Conference on Shape Modeling & Applications 2004*, 181–190.

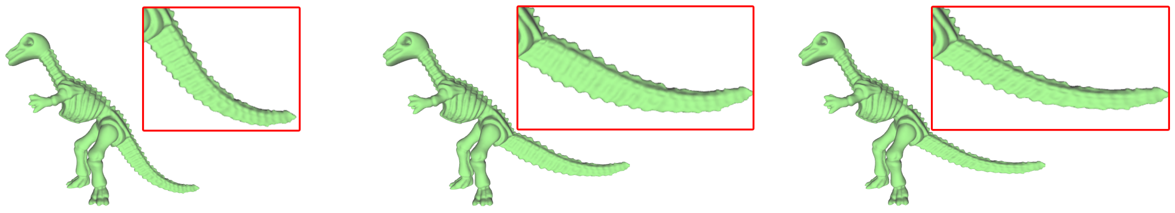
(a) (b) (c)

Figure 4: *Deformation of the dinosaur model: (a) The initial model; (b) The deformed result generated by the vertex-based Laplacian deformation algorithm; (c) The deformed result generated by our edge-based Laplacian deformation algorithm.*

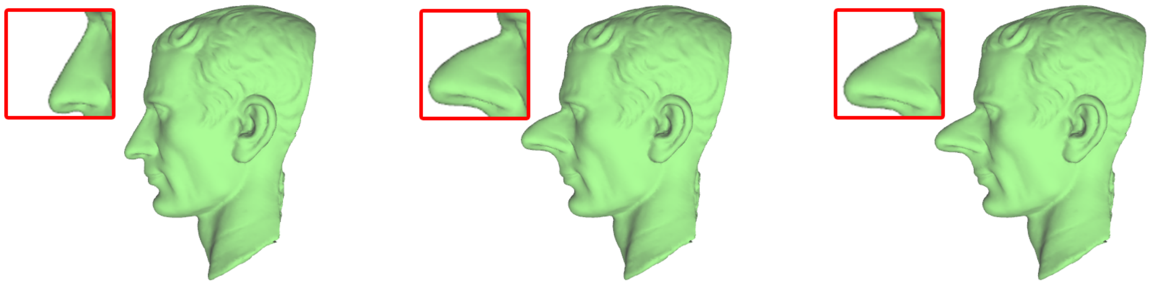
(a) (b) (c)

Figure 5: *Deformation of the Julius Caesar model. (a) The initial model; (b) The deformed result generated by the vertex-based Laplacian deformation algorithm; (c) The deformed result generated by our edge-based Laplacian deformation algorithm.*

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *SGP '07: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 109–116.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H. 2004. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 179–188.

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H. 2004. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: Proceedings of the 31st annual conference on Computer graphics and interactive techniques*, 644–651.

ZAYER, R., RÖSSL, C., KARNI, Z., AND SEIDEL, H. 2005. Harmonic guidance for surface deformation. *Computer Graphics Forum(Proc. Eurographics 2005) 24*, 3, 601–609.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Transactions on Graphics(Proc. SIGGRAPH 2005) 24*, 3, 496–503.