Triangular Mesh Deformation via Edge-Based Graph

Kai Wang[1], Jianmin Zheng[2], Hock-Soon Seah[3] and Yuewen Ma[4]

[1]Nanyang Technological University, wang0452@ntu.edu.sg
[2]Nanyang Technological University, asjmzheng@ntu.edu.sg
[3]Nanyang Technological University, ashsseah@ntu.edu.sg
[4]Nanyang Technological University, mayu0005@e.ntu.edu.sg

ABSTRACT

Mesh deformation is an important mesh editing process which provides a convenient way to modify a given mesh to meet various design requirements. Most of the existing mesh deformation methods establish their formulations in the primal vertex-based domain and are expected to produce globally smooth and consistent deformation results while preserving geometry details as much as possible. In this paper, we present a new surface-based mesh deformation method that performs the computation via an edge-based graph to increase the sampling rate for more accurate shape computation and better deformation results. The user is given the flexibility of adjusting the deformation effect between local shape preservation and global smoothness. Moreover, to simulate the deformation behaviors of regions with different materials, we introduce a stiffness property into the deformation model and present an easy and intuitive way for the user to set the material property. Experimental results demonstrate that our algorithm can produce satisfactory results and make the deformation more flexible.

Keywords: mesh deformation, edge-based graph, material-aware deformation.
DOI: 10.3722/cadaps.2012.345-359

1 INTRODUCTION

Mesh deformation is an effective and convenient method for mesh editing in geometric modeling and computer animation. It allows the user to manipulate one part or the entire surface while preserving some geometric characteristics of the initial surface. It has a wide range of applications in industrial and artistic designs. Due to its importance and popularity, various algorithms have been developed. Though having their respective advantages, those algorithms face various problems, owing to the following considerations in mesh deformation.

First, a triangular mesh is a piecewise linear surface and it is often considered as an approximation of some unknown smooth surface. The approximation effect is related to the number of vertices of the mesh. In general, the larger the number of vertices the mesh contains, the better is the approximation. When a deformation is performed on a surface, the calculation will be conducted on the vertices. The number of vertices determines the sampling rate in approximating a continuous model and thus a denser mesh usually results in more accurate

deformation. However, given a deformation region, the number of vertices is fixed and it is not trivial to have more points be involved in the deformation calculation.

Second, in the deformation process the local shape feature is often required to be well preserved and the deformed shape should be natural and smooth to satisfy aesthetics or other requirements. A natural problem then arises: what is a good balance between the rigidity and fairness? The answer is subjective. A satisfactory editing result depends on the algorithm adopted, the specific models and the desired effect as well.

Third, an object in real world may be composed of different materials, which may exhibit different behaviors when the object is deformed. For example, with different stiffness properties of the materials, different deformation effects between rigidity and smoothness will appear under the deformation. Most of the existing deformation methods are purely geometrically-based and they ignore the material properties of the object.

To solve these problems and make the deformation more flexible, we propose a flexible mesh deformation algorithm which performs the computation through an edge-based graph. Instead of formulating the deformation on the primal domain, we build an edge-based graph and carry out the deformation computations on the graph, which involves more nodes and thus produces better deformation results. We then propose a mixed energy model that combines the first order and the second order discrete differential quantities to balance local shape preservation and smoothness. The user can adjust the values of the balance parameters to control the deformation results. Moreover, these introduced balance parameters are used to reflect the stiffness property of the object material and thus make the deformation algorithm material-aware. A simple sketching tool is provided to specify the stiffness property. The novelty of the paper thus lies in the use of the edge-based graph for the deformation formulation and the use of balance parameters for controlling the stiffness and producing various reasonable deformation results in a mesh deformation task.

The rest of the paper is structured as follows: Section 2 reviews some existing mesh deformation algorithms. Section 3 presents our edge-based mesh deformation algorithm. Section 4 provides experimental results and discussions. Section 5 concludes the paper.

2 RELATED WORK

There has been extensive research on mesh deformation and many methods have been developed. Existing methods can roughly be divided into two categories: the space deformation [9] and the surface deformation [4]. The space deformation deforms a mesh by warping its surrounding space through manipulating control objects, such as a control lattice [15] or cage [10],[11]. It has the merit of being computationally efficient and applicable to a variety of object representations. However, it is not direct or intuitive for interactive design. The surface deformation manipulates the mesh surface directly. It usually requires the user to specify some vertices on the mesh and their new positions. These vertices are called the *handle* vertices. The user also needs to specify the *region-of-interest (ROI)*, which deforms with the handle vertices in the deformation. The rest of the vertices that are outside of the ROI remain unchanged, so they are called *static* vertices. Usually the vertices within the ROI are deformed by some rules such as minimizing a certain objective function.

Sorkine et al. [16] proposed a surface deformation method that attempts to preserve the first order differential vectors of the mesh surface. Thus the objective function is a function of the change of the first order differential vectors at the vertices. As a result, the method works well for preserving the local details of the mesh. However, the deformed shape usually appears too rigid and the smoothness is not well guaranteed especially when stretching or shrinking occurs.

Many deformation methods were proposed to preserve Laplacian vectors at vertices to achieve smooth deformation. The Laplacian vector at a vertex is a vector that points from the weighted center of the 1-ring neighborhood to the vertex itself. It is related to the second order differential property of the mesh [1]. Usually Laplacian based deformation methods can yield globally natural and smooth results. However, since Laplacian vectors are not invariant under rotation and scale

transformations, additional steps or modifications are needed to estimate the rotation and scaling in the deformation. Typically there are two approaches for this purpose: explicit and implicit.

The explicit approach estimates the rotation and scaling transformations by interpolating the transformations at the handles according to geodesic distances [19],[21] or harmonic functions [20],[14]. This is a linear problem, which can be solved very quickly. However, the transformations at the handles need to be specified manually, which is not accurate or convenient. What's more, the rotation of the ROI vertices caused by the translation of the handles cannot be detected (which is known as the translation-insensitivity problem). Thus the local details cannot be well preserved under this circumstance.

The implicit approach derives the local transformations according to the deformed and undeformed vertex positions. This actually is a "chicken-and-egg" problem since the computation of the deformed vertices also depends on the transformations. [12] proposed to establish local frames at the undeformed and deformed vertices and estimated the transformation based on the two frames. [17] simplified the rotation matrix as a linear one and performed the calculation by solving a large linear system. [8] computed an affine transformation at each vertex and then extracted the rotation and uniform scale matrices as the local transformation. However, none of them are accurate since they just gave an approximation of the local transformations.

There are also some methods that consider both the first and second order differential quantities [7],[6]. They use the inner angles and principal curvatures and adopt a nonlinear solver to minimize the energy function for deformation. Although both local shape preservation and global smoothness could be obtained, the slow computation makes them difficult to meet the requirement of interactive design.

In all the above-mentioned methods, the calculations are implemented directly in the primal domain. Au et al. [3] extended the Laplacian approach into the dual domain. By making use of the advantage of the dual mesh, shearing could be avoided and more constraints are imposed. Nevertheless, their iterative method is slow in convergence and shape preservation sometimes cannot be achieved due to the limitation of the Laplacian coordinates.

While most deformation methods assume that the mesh surface is composed of uniform materials, the work in [14] used a simple scalar parameter to define the stiffness of the material and control the rotation and scale of local regions on the surface. It can simulate the deformation of a mesh with non-uniformly distributed materials, but suffers the same problems as the explicit methods for calculating the local transformations of the Laplacian coordinates.

3 FLEXIBLE MESH DEFORMATION VIA EDGE-BASED GRAPH

In a mesh deformation process, the user specifies the change of the handle and then the algorithm determines the deformation of the vertices in the ROI. Our basic idea is to deform the vertices in the ROI such that a mixed energy functional is minimized and this minimization problem is formulated on an edge-based graph. The mixed energy functional includes the change of the first order and second order differential quantities that reflect stretching and bending. Based on Euler-Poincare formula, the number of edges on a triangular mesh is approximately three times that of vertices. Thus the energy functional formulated on the edge-based graph better approximates the continuous version of the energy functional of the smooth surface than that on the primal domain. Thus the formulation on the edge-based graph tends to render better deformation results, especially when rigid or smooth deformation effects are required. See Fig.1 for an example. In the following we will describe our deformation algorithm.

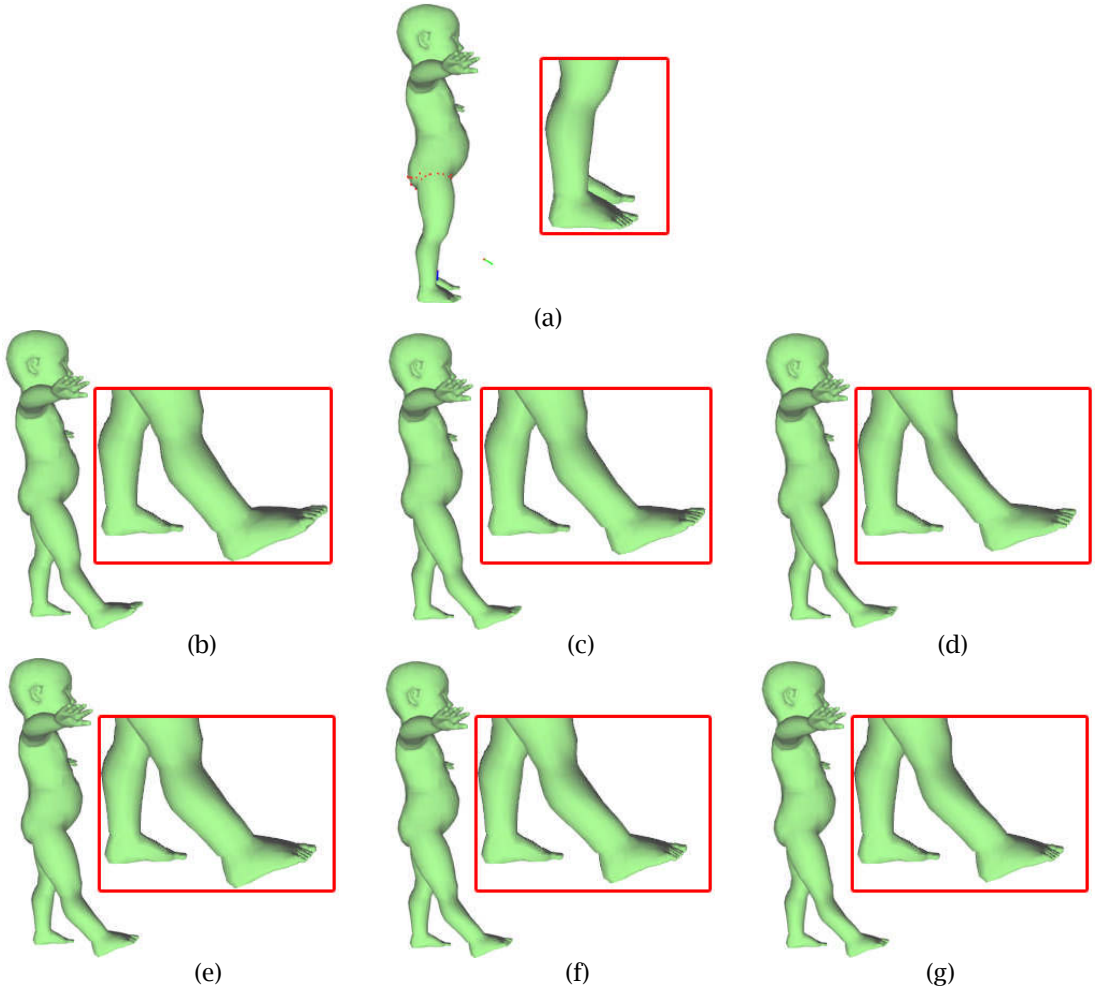


Fig. 1: Flexible deformation of the *Baby* model. (a) The initial model. (b)~(d) Results of the flexible deformation algorithm performed on the primal domain, with the global balance parameters valued 0, 0,5 and 1.0 respectively. (e)~(g) Results of the flexible deformation algorithm performed via the edge-based graph, with the global balance parameters valued 0, 0,5 and 1.0 respectively.

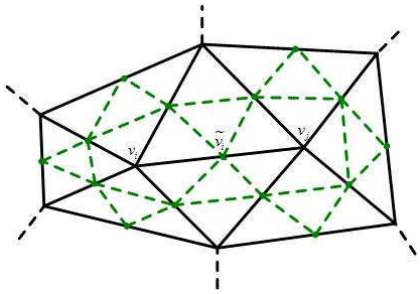


Fig. 2: An illustration of an edge-based graph. The black lines represent the primal mesh and the green lines form the edge-based graph.

3.1 Edge-Based Graph

Let $M=(V,E)$ be a triangular mesh with n vertices. $V=\{v_i \mid v_i \in R^3, i=1,...,n\}$ denotes the set of vertices and $E=\{e_{ij}=(v_i,v_j) \mid v_i,v_j \in V, i \neq j\}$ represents the set of edges. Correspondingly, we define $\widetilde{M}=(\widetilde{V},\widetilde{E})$ as the corresponding edge-based graph for M, where $\widetilde{V}=\{\widetilde{v_i} \mid \widetilde{v_i} \in R^3, i=1,...,n_e\}$ and $\widetilde{E}=\{\widetilde{e_{ij}}=(\widetilde{v_i},\widetilde{v_j}) \mid \widetilde{v_i},\widetilde{v_j} \in \widetilde{V}, i \neq j\}$ are the sets of nodes and edges of the edge-based graph respectively. Specifically, a node $\widetilde{v_i}$ is defined as the midpoint of the edge e_{ij} in the primal domain: $\widetilde{v_i}=(v_i+v_j)/2$. n_e represents the number of nodes in the edge-based graph, which equals the number of edges on the primal mesh M. Each node $\widetilde{v_i}$ corresponding to primal edge e_{ij} is connected to four neighboring nodes corresponding to the four primal edges that share the vertices of e_{ij}. In this way, the valence of each node in the edge-based graph is always four. The conversion from the primal vertex positions V to the edge-based graph vertex positions $\widetilde{V}$ can be accomplished by an $n_e \times n$ sparse matrix D. Each row of D has two non-zero elements that are 1/2. Fig. 2 shows an edge-based graph constructed from a primal mesh.

3.2 Flexible Mesh Deformation

As has been introduced in Section 2, most deformation algorithms aim to preserve either the first or the second order discrete differential quantities of the mesh surface and thus can produce only one kind of deformation result, which may be either too rigid or smooth. Meanwhile, it is not easy to give a good balance between the rigidity and smoothness of a deformation.

To solve these problems, we propose a method which considers both of the two differential quantities and balance between the rigidity and smoothness of the deformation in a flexible manner. We also formulate it as a linear system such that the computation can be fast enough for interactive design.

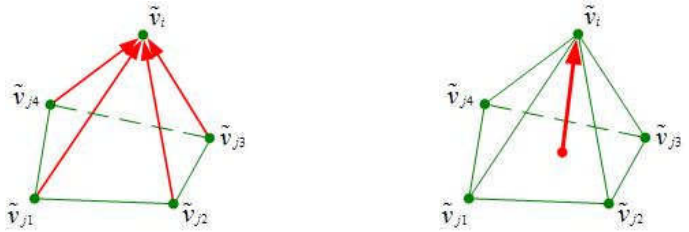
3.2.1 First and second order discrete differential quantities

First of all we need to find proper representations for the first and second order discrete differential quantities of the mesh surface.

For a smooth surface, the first order differential properties are usually represented by length, area, etc. However, these will make the system nonlinear. So for triangular meshes we choose to use edge vectors at each node, which we called the 1-form, to represent the first order differential structures of the mesh surface. The 1-form contains the information of edge lengths and can help to make the final system a linear one.

Specifically, for a node $\widetilde{v_i}$, the 1-form can be represented by the edge vectors $\widetilde{e_{ij}}$ which start from the one-ring neighborhood $\widetilde{v_j},\widetilde{e_{ij}} \in \widetilde{E}$ and point to $\widetilde{v_i}$, as shown in Fig. 3(a). It can totally decide the shape of the local area around a node. That is to say, as long as the edge vectors connecting $\widetilde{v_i}$ and its neighbors are fixed, the shape of the local area around $\widetilde{v_i}$ is fixed.

The second order differential properties of a smooth surface are usually represented by curvatures, which also make the system a nonlinear one [7]. The Laplacian vector provides a proper choice for representing the second order differential properties of the mesh surface, since its magnitude approximates the mean curvature at each vertex and the final system formed is a linear one. So we extend the traditional definition of Laplacian vector from the primal mesh to the edge-based graph and use it to represent the second order discrete differential quantity of the mesh surface, as shown in Fig. 3(b).



(a) (b)

Fig. 3: (a) The 1-form. (b) The Laplacian vector.

(a) (b)

Fig. 4: The Laplacian vector $\tilde{\delta}_i$ in the edge-based graph and the dihedral angle α_i between adjacent faces in the primal domain. (a) The shorter $\tilde{\delta}_i$ is, the larger α_i will be. (b) The longer $\tilde{\delta}_i$ is, the smaller α_i will be.

Mathematically, the Laplacian vector $\tilde{\delta}_i$ at $\tilde{v}_i$ is defined as:

$$\tilde{\delta}_i = L(\tilde{v}_i) = \sum_{j \in N(i)} \tilde{\omega}_{ij}(\tilde{v}_i - \tilde{v}_j), \tag{3.1}$$

where $N(i) = \{j \mid \tilde{e}_{ij} \in \tilde{E}\}$ is the index set of adjacent nodes in the neighboring ring of $\tilde{v}_i$ and $\tilde{\omega}_{ij}$ is the weight of the edge $\tilde{e}_{ij}$. Here we use the cotangent weight scheme [13] to compute $\tilde{\omega}_{ij}$. In fact, $L(\cdot)$ can be regarded as the discretized Laplace-Beltrami operator with cotangent coefficients on triangular meshes [4].

It should be noted that the Laplacian vector $\tilde{\delta}_i$ in the edge-based graph can also be regarded as a measurement of the dihedral angle α_i of two adjacent triangles in the primal mesh, which also represents the second order differential property of the mesh surface. The magnitude of $\tilde{\delta}_i$ is inversely proportional to the size of α_i, as can be seen in Fig. 4.

These two representations of discrete differential quantities encode the relative position of each node w.r.t. its neighboring nodes and represent the local geometry properties of the mesh surface. Next we will introduce how to combine them to get a satisfactory deformation result flexibly.

3.2.2 Energy function for flexible deformation

The approach for solving the deformation problem of the surface-based method is to designate the positional constraints for the handle and static vertices and solve for the positions of the ROI vertices by fitting the local geometry properties of the deformed mesh to those of the initial mesh. The properties we want to preserve include both the first and second order discrete differential

properties, which are represented by the 1-form and Laplacian vectors.

Since the 1-form at each node $\widetilde{v_i}$ can totally decide the shape of a local neighborhood area, it is considered as defining the local shape of the mesh surface in a unique manner. Mathematically, to preserve the 1-form of the initial mesh, the energy we try to minimize for each node $\widetilde{v_i}$ is:

$$E(\widetilde{v_i}') = \sum_{j \in N(i)} \widetilde{\omega_{ij}} \, || \, \widetilde{e_{ij}}' - R_i \widetilde{e_{ij}} \, ||^2, \tag{3.2}$$

where $\widetilde{v_i}'$ is the new position of $\widetilde{v_i}$, $\widetilde{e_{ij}} = \widetilde{v_i} - \widetilde{v_j}$ and $\widetilde{e_{ij}}' = \widetilde{v_i}' - \widetilde{v_j}'$ are the undeformed and deformed edge vectors between $\widetilde{v_i}$ and $\widetilde{v_j}$ respectively. $\widetilde{\omega_{ij}}$ is the same weight as that in Eqn. (3.1). R_i represents the rotation matrix associated to $\widetilde{v_i}$. The optimal rotation matrix R_i could be calculated given the old and new values of the edge vectors. Detailed deduction can be found in [16].

The rigid transformation of the 1-form during deformation strives to do a *hard* transform to the mesh. We say *hard* as the edge vectors of each node can completely determine the shape of its neighborhood area. As long as each vector keeps fixed, the shape of this area will not change at all.

To preserve each Laplacian vector of the mesh surface during deformation, we try to minimize its change up to an affine transformation T_i. Since shearing and anisotropic scale will lead to distortions of the local features [8], we assume T_i be composed of rotation and isotropic scale, i.e. $T_i = s_i R_i$. s_i and R_i represent the scale coefficient and rotation matrix separately. The energy to minimize is:

$$E(\widetilde{v_i}') = || \sum_{j \in N(i)} \widetilde{\omega_{ij}} (\widetilde{v_i}' - \widetilde{v_j}') - T_i \sum_{j \in N(i)} \widetilde{\omega_{ij}} (\widetilde{v_i} - \widetilde{v_j}) \, ||^2 . \tag{3.3}$$

The deformation of preserving the Laplacian vectors is a relatively *soft* deformation, comparing to the hard one, since the Laplacian vector cannot exactly represent the shape of the local area around each node. The word *soft* has another meaning that the deformed shape tends to be smooth, since the change of the second order differential quantity is minimized to prevent the sharp change of the mesh shape within a local area.

Up to now, we have obtained two linear methods for mesh deformation by preserving the 1-form and Laplacian vectors respectively. The unknowns in the two systems are both the set of deformed node positions. However, neither of the two methods can satisfy our requirement of obtaining various deformation effects between rigidity and smoothness. In the first method, while the representation of the local shape is quite accurate, sometimes the deformation will be too rigid that smoothness cannot be obtained. Besides, a definite rigid transformation means that scale is not considered, and thus some local features cannot be preserved under this kind of transformation. In the second method, since the Laplacian vector cannot exactly represent the shape of a local region, local shape preservation may not be achieved. However, the deformation calculated using this method is relatively smooth. Moreover, scale transformation is considered in the computation to make the result more natural and help to preserve the local features under stretching or shrinking.

To make use of the advantages of these two methods and give the user flexibilities to choose the most desirable deformation effect between rigidity and smoothness subject to a specific model, we propose a new algorithm which combines the two methods using a balance parameter λ. When $\lambda = 1$, only the first order discrete differential quantity of each node is preserved and the deformation is the most rigid; when $\lambda = 0$, only the second order discrete differential quantities are preserved and a smooth result is achieved. Multiple intermediate results can be obtained by setting different λ values between 0 and 1.

Considering the positional constraints $\{\widetilde{c_i}\}, i \in \{m, ..., n\}, m < n$, the overall energy we aim to minimize for a mesh deformation task with n nodes involved is:

$$E(V') = \sum_{i=1}^{n} \lambda_i \Big(\sum_{j \in N(i)} \widetilde{\omega_{ij}} \, \| \widetilde{e_{ij}}' - R_i \widetilde{e_{ij}} \|^2 \Big) + \sum_{i=1}^{n} (1-\lambda_i) \Big\| \sum_{j \in N(i)} \widetilde{\omega_{ij}} (\widetilde{v_i}' - \widetilde{v_j}') - T_i \sum_{j \in N(i)} \widetilde{\omega_{ij}} (\widetilde{v_i} - \widetilde{v_j}) \Big\|^2 + \sum_{i=m}^{n} \| \widetilde{v_i}' - \widetilde{c_i} \|^2, \tag{3.4}$$

in which the first and second items of the right hand side are the energies measured by the change of the 1-form and Laplacian vectors and the third item is the positional constraints during deformation. The balance parameter λ_i ($0 \le \lambda_i \le 1$) is used to control the rigidity and smoothness of the deformation at node $\widetilde{v_i}$.

It is not difficult to see that the definitions of 1-form and Laplacian vectors and the flexible deformation algorithm can be easily transferred to the primal domain, by replacing the variables defined in the edge-based graph in Eqn. (3.4) with those of the primal domain. This is useful for comparing the deformation results calculated directly in the primal domain and via the edge-based graph. Detailed comparisons can be found in Section 5.

3.2.3 Solving of the system

To solve this system, we use an iterative method which can be outlined as follows.

First we build an edge-based graph for the deformation region of the mesh and then compute the partial derivatives of the first and second items in energy function 3.4 w.r.t. the unknown node positions $\widetilde{V}'$ and get two linear systems $A_1 \widetilde{V}' = b_1$ and $A_2 \widetilde{V}' = b_2$. An important observation is that if the coefficients λ_i and $1-\lambda_i$ are not considered, A_1 and A_2 happen to be the same. So only one of them needs to be constructed in the beginning and we just need to focus on the calculation of the right hand side of the system in each iteration.

Then an initial deformation result is estimated using the naive Laplacian deformation [1] which does not consider rotation and scale transformations of the Laplacian vectors. Based on the estimated vertex positions, we can compute the rotation matrix R_i and the transformation matrix T_i. Since the optimal R_i can be uniquely calculated through singular value decomposition [16], we make use of it to compute the optimal scale coefficient s_i, such that $T_i = s_i R_i$ can be obtained. This way, computations of R_i and T_i are associated and the overall calculation process is accelerated. The right hand side of the system is then updated by multiplying the calculated R_i and T_i and the system can be solved.

Next some iterations are performed for recomputing R_i and T_i, updating the right hand side of the system and solving it. This process continues until convergence.

As all the above computations are performed in the edge-based graph, we need to reconstruct the primal mesh by using the conversion matrix D introduced in Section 3.1 as well as the positional constraints for primal handle and static vertices. However, since the position of each node in the edge-based graph can be represented as the linear combination of two primal vertex positions, an easier way to solve the system is to integrate D into Eqn. (3.4) and directly take the primal vertex positions as unknowns.

In the above steps we assume that the balance parameters have been predefined and treat them as constants during the calculation. Next we will introduce the methods for specifying them and the corresponding significance.

3.2.4 Specification of the balance parameters

We provide two methods for specifying the balance parameters: the global method and the local method.

In the global method, we let the balance parameter λ_i for each node $\widetilde{v_i}$ equal to a same value λ, such that energy function Eqn. (3.4) turns to:

$$E(V') = \lambda\sum_{i=1}^{n}\Big(\sum_{j\in N(i)}\widetilde{\omega_{ij}}\,\|\,\widetilde{e_{ij}'} - R_i\widetilde{e_{ij}}\,\|^2\Big) + (1-\lambda)\sum_{i=1}^{n}\Big\|\sum_{j\in N(i)}\widetilde{\omega_{ij}}(\widetilde{v_i}' - \widetilde{v_j}') - T_i\sum_{j\in N(i)}\widetilde{\omega_{ij}}(\widetilde{v_i} - \widetilde{v_j})\Big\|^2 + \sum_{i=m}^{n}\|\,\widetilde{v_i}' - \widetilde{c_i}\,\|^2. \tag{3.5}$$

This global method provides a proper choice when the user wants to make the entire editing region of a mesh deform in a consistent manner. The rigidity and smoothness of the deformation can be well balanced through setting a proper λ value between 0 and 1. The user can select a proper λ value to produce a satisfactory result, according to the specific model and the deformation effect he needs. Generally speaking, larger λ values make the deformation more rigid and smaller λ values make it more fair. It can be seen that previous methods which preserve either the first or the second order differential properties of the mesh surface is just a special case of Eqn. (3.5).

Examples of the global method can be seen in Fig. 5-7.

Although the global method provides an efficient way of calculating a natural and consistent deformation result for the entire editing region, it ignores the individual property of each local area on the mesh surface and sometimes is not flexible. So we also provide a local method for the specification of the balance parameters.

Since objects in real world are often composed of non-uniformly distributed materials, their deformation behaviors vary from rigidity to smoothness across the surface, depending on the stiffness of the materials on each local region. However, current mesh deformation algorithms either ignore the underlying materials of the mesh surface or suffer various problems. The physics or skeleton based algorithms provide feasible approaches to solve this problem. Nevertheless, they are either slow in computation or limited to a subset of models [14].

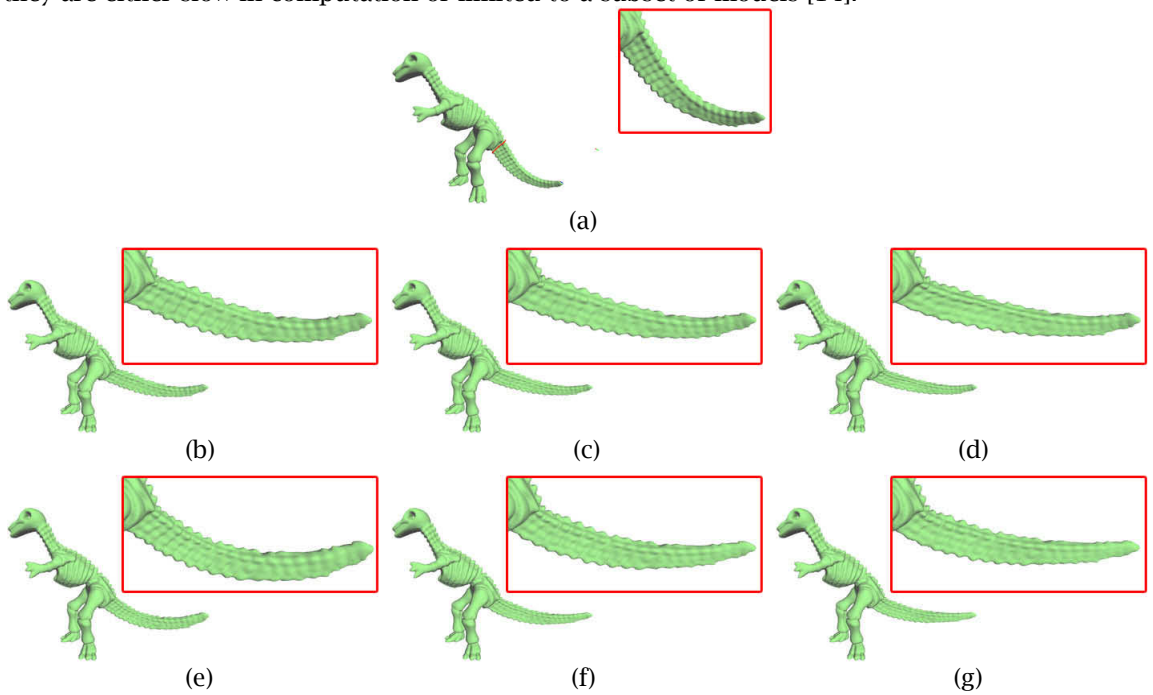


Fig. 5: Flexible deformation of the *Dinosaur* model. (a) The initial model. (b)~(d) Results of flexible deformation algorithm performed on the primal domain, with the global balance parameters valued 0, 0,5 and 1.0 respectively. (e)~(g) Results of flexible deformation algorithm performed via the edge-based graph, with the global balance parameters valued 0, 0,5 and 1.0 respectively.

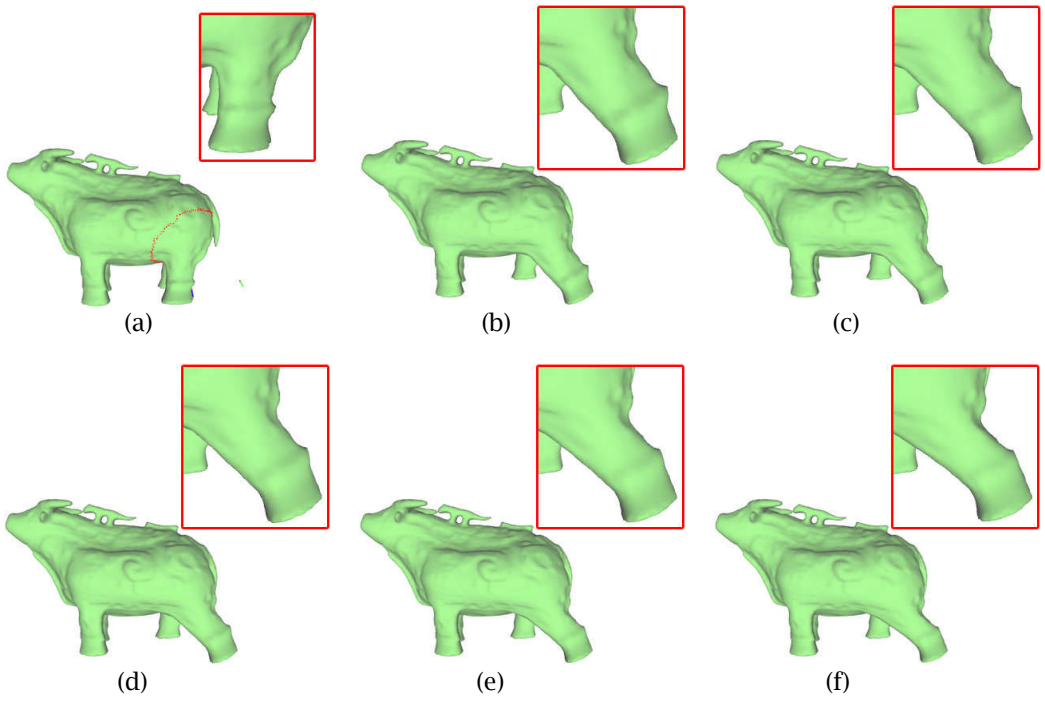


Fig. 6: Flexible deformation of the *Buffle* model via the edge-based graph under different balance parameter values. (a) The initial model. (b)~(f) Deformation results with the global balance parameters valued 0, 0.2, 0.5, 0.8, 1.0 respectively.

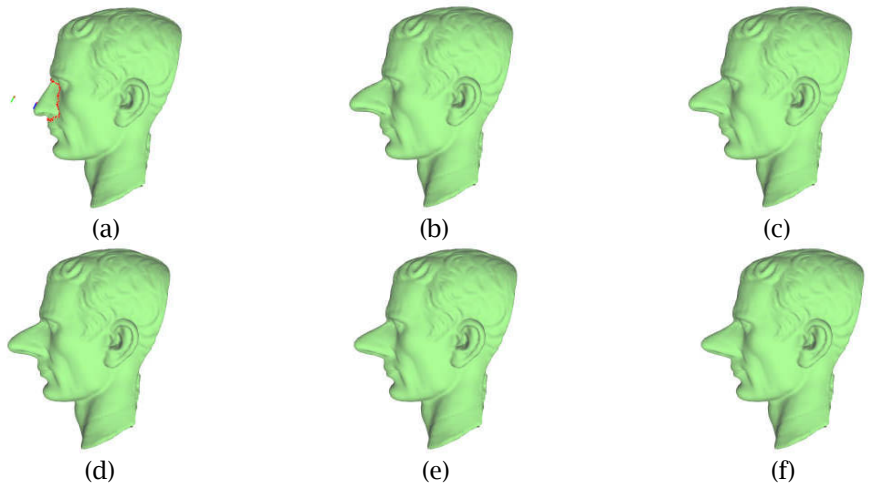


Fig. 7: Flexible deformation of the *Julius—Caesar* model via the edge-based graph under different balance parameter values. (a) The initial model. (b)~(f) Deformation results with the global balance parameters valued 0, 0.2, 0.5, 0.8, 1.0 respectively.

Since the balance parameter can be used to control the rigidity and smoothness of the deformation, we can use it to define the stiffness property of the material on each local area of the mesh surface, such that the regions with different materials will have distinguished behaviors in a

same deformation task. In this case, λ_i for each $\widetilde{v_i}$ in Eqn. (3.4) should be defined separately according to the material property of $\widetilde{v_i}$. The larger λ_i is, the stiffer the material will be and the deformation at $\widetilde{v_i}$ should be relatively rigid; otherwise, the material will be softer and the deformation should be smooth.

Here we provide sketching tools, which are intuitive and flexible for interactive designs, for the user to designate materials on different parts of the mesh surface. The user can first set a scalar value which represents the stiffness property of the material and then use lasso or painting brush tools to specify the material for the selected part. Since the entire user operations are performed on the surface of the primal mesh while the calculations are implicitly implemented on the edge-based graph, after the materials of the primal vertices are specified, we calculate the material for each node $\widetilde{v_i}$ of the edge-based graph by averaging those of the two end vertices of the primal edge e_{ij} that $\widetilde{v_i}$ lies on.

Results of the local method can be seen in Fig. 8-10.

It can be summarized that when a globally consistent deformation is desired, the global method can be used to produce a well-balanced result between rigidity and smoothness, subject to the specific model and user requirement; when the user wants to compute the deformation of a mesh surface with non-uniformly distributed materials, then the local method can be used to specify the stiffness properties of the materials on the local regions and get a natural and realistic result.

4 RESULTS AND DISCUSSIONS

We have implemented our deformation algorithm using C++ on an Intel Xeon 2.27G computer with 2GB RAM. To solve the linear equations, the LAPACK library [2] is used. For the operations of the mesh deformation, we adopted a user interface presented in [18], which allows the user to sketch strokes on the mesh surface and on a reference plane as the handle curve and its deformed shape (indicated by the blue and green curves in all the examples) respectively. The ROI is also specified through sketching. The red dots in the examples indicate the static vertices.

Figs. 1 and 5 compare the deformation results of the flexible deformation algorithm performed directly in the primal domain and via the edge-based graph. In Fig. 1 we can see that when λ equals 0 or 1, which means a smooth or rigid deformation is requested, calculations in the primal domain produce poor results: when $\lambda = 0$, the shape of the foot is not well preserved and when $\lambda = 1$, distortions appear around the knee part. However, the results calculated via the edge-based graph are much better. Similar observations can be found in Fig. 5, where the tail of the dinosaur is lifted and stretched to some extent.

In Fig. 6 we stretch the leg of the model and present different results under various λ values. The result of $\lambda = 0$ is quite smooth, but the leg becomes fatter than the initial one, which means shape preservation is not satisfactory. The result of $\lambda = 1$ preserves the shape well. However, it lacks some smoothness. The result of $\lambda = 0.5$ seems to be the most natural and realistic one.

Sometimes the most satisfactory result cannot be easily identified, such as in the example shown in Fig. 7, in which we deform the nose of a head model. This is reasonable because besides the requirement of local feature preservation and global smoothness, the best deformation also depends on the individual user preference. From this perspective, all these results are acceptable and the best is left for the user to choose.

It is also observed that for some models, the differences between the deformation results produced through calculating directly in the primal domain and via the edge-based graph are not quite obvious when $\lambda = 0.5$. This is because the result of $\lambda = 0.5$ can be regarded as a high-level interpolation of those of $\lambda = 0$ and $\lambda = 1$. Even the results of the pure rigid and smooth deformations done in the primal domain are far from satisfactory, interpolation of them may

reduce the distortion to some extent. However, the capability of producing multiple natural and realistic results under various λ values makes the edge-based method more flexible.

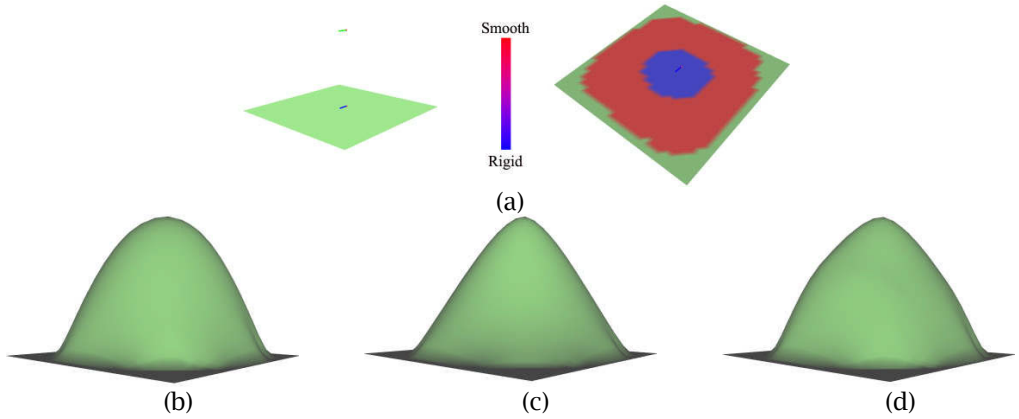


Fig. 8: Material-constrained deformation result of the *plane* model. (a) The initial model and its specified materials. (b)~(c) Deformation results of uniformly distributed materials with global balance parameter valued 0 and 1 respectively. (d) Deformation result with the specified non-uniform materials.

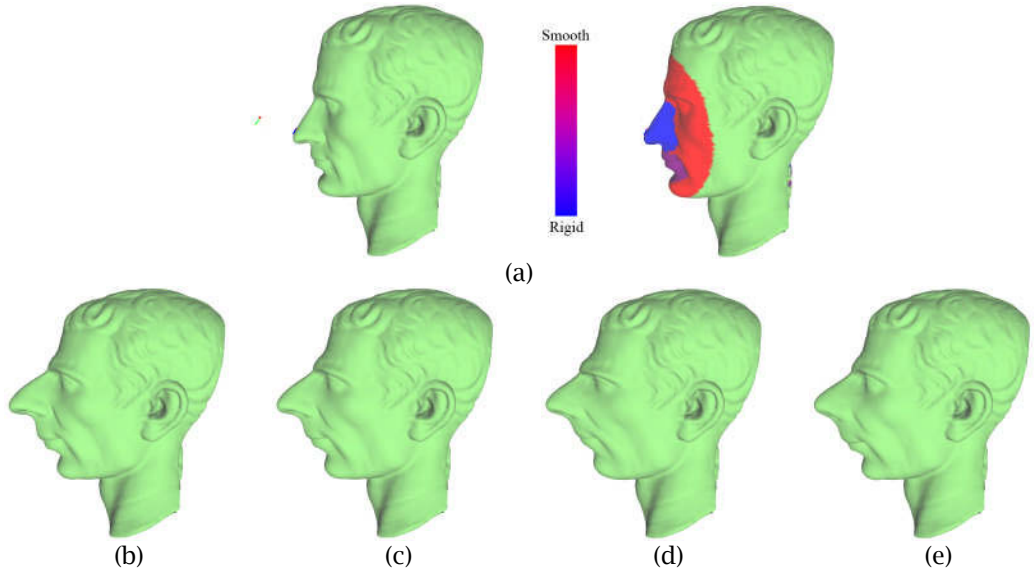


Fig. 9: Material-constrained deformation of the *Julius—Caesar* model. (a) The initial model and its specified materials. (b)~(d) Deformation results of uniformly distributed materials with global balance parameter valued 0, 0.5 and 1 respectively. (e) Deformation result with the specified non-uniform materials.

Figs. 8, 9 and 10 show the results of material-constrained mesh deformation. The deformation results for the same model with certain uniform materials are also provided for comparisons. In Fig. 8, the material at the center part of the plane is set to be the stiffest and the material in the surrounding part to be soft. As can be seen, the deformations of these two parts approximate those of the models with corresponding material properties. In Fig. 9, we define different

materials on the nose, mouth and face, in order to let the nose deform rigidly, the face change smoothly, and the mouth change in an in-between effect. The result looks as if the nose, mouth and face are transplanted from the models with corresponding uniform materials. Similar results can be observed on the head, ear and body of the pig in Fig. 10, which is an exaggerated deformation effect often seen in cartoons.

It can be seen that using the material-constrained deformation algorithm, we can make the deformation more flexible to produce different natural and realistic deformation results, which is usually difficult using previous methods. Comparing with the method proposed in [14], our approach does not require the user to manually specify the local transformations of the handles and avoids the translation-insensitivity problem during deformation.

It is worth noting that our algorithm involves a larger linear system due to the use of the edge-based graph and thus the computation takes more time. However, in our experiments we found that only up to three or four iterations are needed to reach convergence, and since only the right hand side of the linear system needs to be updated at each iteration, the computation is still more stable and faster than those nonlinear methods.

5 CONCLUSION

This paper has proposed to compute mesh deformation through an edge-based graph. The edge-based graph enables a higher sampling rate in shape computation and thus produces better deformation effects especially for coarse mesh models. A mesh deformation model, which combines both the first order and the second order differential quantities, has been proposed. Thus the flexibility of adjusting the deformation results between rigidity and smoothness is obtained. Furthermore, the introduced parameters in the deformation model can be used to control the stiffness property of the surface material. With a simple tool to set the material property, the deformation effects of a mesh with non-uniformly distributed materials can easily be achieved. Experimental results have demonstrated the usefulness of the edge-based graph in mesh deformation and the flexibilities of the proposed deformation algorithm.

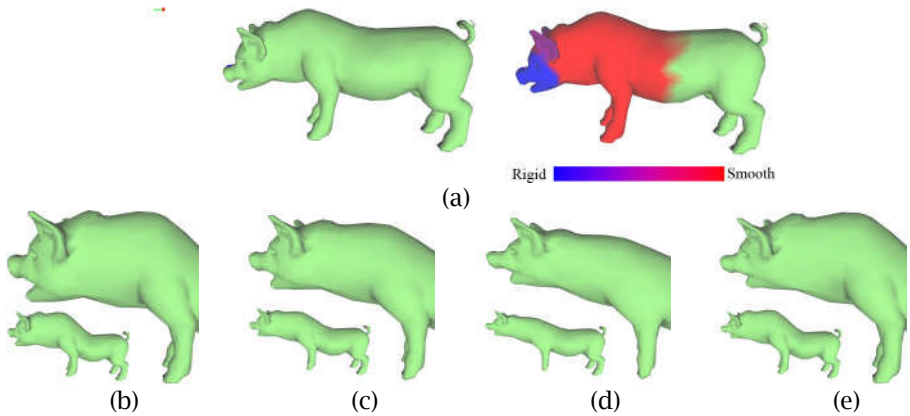


Fig. 10: Material-constrained deformation of the *pig* model. (a) The initial model and its specified materials. (b)~(d) Deformation results of uniformly distributed materials with global balance parameter valued 0, 0.5 and 1 respectively. (e) Deformation result with the specified non-uniform materials.

6 ACKNOWLEDGMENT

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

[1] Alexa, M.: Local control for mesh morphing, Proceedings of the International Conference on Shape Modeling & Applications (SMI '01), 2001, 2–9.(DOI:10.1109/SMA.2001.923392)

[2] Anderson, E.; Bai, Z.; Bischof, C.; Demmel, J.; Dongarra, J.; Croz, D.; Greenbaum, A.; Hammarling, S.; Ckenney, A.; Ostrouchov, S.; Sorensen, D.: LAPACK's user's guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[3] Au, O.; Tai, C.; Liu, L.; Fu, H.: Dual Laplacian editing for meshes, IEEE Transaction on Visualization and Computer Graphics, 12(3), 2006, 386–395.(DOI:10.1109/TVCG.2006.47)

[4] Botsch, M.; Sorkine, O.: On linear variational surface deformation methods, IEEE Transactions on Visualization and Computer Graphics, 14(1), 2008, 213–230. (DOI:10.1109/TVCG.2007.1054)

[5] Do Carmo, M.: Differential geometry of curves and surfaces, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[6] Eigensatz, M.; Pauly, M.: Positional, metric, and curvature control for constraint-based surface deformation, Computer Graphics Forum (Proc. Eurographics 2009), 28(2), 2009, 551–558. (DOI:10.1111/j.1467-8659.2009.01395.x)

[7] Eigensatz, M.; Sumner, R.; Pauly, M.: Curvature-domain shape processing, Computer Graphics Forum (Proc. Eurographics 2008), 27(2), 2008, 241–250. (DOI: 10.1111/j.1467-8659.2008.01121.x)

[8] Fu, H.; Au, O.; Tai, C.: Effective derivation of similarity transformations for implicit Laplacian mesh editing, Computer Graphics Forum, 26(1), 2007, 34–45. (DOI: 10.1111/j.1467-8659.2007.00940.x)

[9] Gain, J.: Bechmann D.: A survey of spatial deformation from a user-centered perspective, ACM Transactions on Graphics, 27(4), 2008, 1-21. (DOI:10.1145/1409625.1409629)

[10] Ju, T.; Schaefer, S.; Warren, J.: Mean value coordinates for closed triangular meshes, Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05), 2005, 561–566. (DOI:10.1145/1073204.1073229)

[11] Lipman, Y.; Levin, D.; Cohen-Or, D.: Green coordinates, ACM Transactions on Graphics (Proc. SIGGRAPH 2008), 27(3), 2008, 1–10. (DOI:10.1145/1360612.1360677)

[12] Lipman, Y.; Sorkine, O.; Cohen-Or, D.; Levin, D.; Rossl, C.; Seidel, H.: Differential coordinates for interactive mesh editing, Proceedings of the International Conference on Shape Modeling & Applications (SMI '04), 2004, 181–190. (DOI:10.1109/SMI.2004.30)

[13] Meyer, M.; Desbrun, M.; Schroder, P.; Barr, A.; Discrete differential-geometry operators for triangulated 2-manifolds, Visualization and Mathematics III, 2002, 35–57.

[14] Popa, T.; Julius, D.; Sheffer, A.: Material-aware mesh deformations, Proceedings of the International Conference on Shape Modeling & Applications (SMI '06), 2006, 22. (DOI:10.1145/1186954.1186960)

[15] Sederberg, T.; Parry, S.: Free-form deformation of solid geometric models, ACM SIGGRAPH Computer Graphics, 20(4), 1986, 151–160. (DOI:10.1145/15886.15903)

[16] Sorkine, O.; Alexa, M.: As-rigid-as-possible surface modeling, Proceedings of the 2007 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '07), 2007, 109–116.

[17] Sorkine, O.; Cohen-Or, D.; Lipman Y.; Alexa M.; Rossl C.; Seidel H.: Laplacian surface editing, Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing (SGP '04), 2004, 179–188. (DOI:10.1145/1057432.1057456)

[18] Wang, K.; Zheng, J.; Seah, H.: Reference Plane assisted sketching interface for 3d freeform shape design, Proceedings of International Conference on Cyberworlds (CW '10), 2010, 105-112. (DOI:10.1109/CW.2010.23)

[19] Yu, Y.; Zhou, K.; Xu, D.; Shi, X.; Bao, H.; Guo, B.; Shum, H.: Mesh editing with poisson-based gradient field manipulation, Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04), 2004, 644–651. (DOI:10.1145/1186562.1015774)

[20] Zayer, R.; Rossl, C.; Karni, Z.; Seidel, H.: Harmonic guidance for surface deformation, Computer Graphics Forum (Proc. Eurographics 2005), 24(3), 2005, 601–609. (DOI: 10.1111/j.1467-8659.2005.00885.x)

[21] Zhou, K.; Huang, J.; Snyder, J.; Liu, X.; Bao, H.; Guo, B.; Shum, H.: Large mesh deformation using the volumetric graph laplacian, ACM Transactions on Graphics (Proc. SIGGRAPH 2005), 24(3), 2005, 496–503. (DOI:10.1145/1186822.1073219)