Interactive Mesh Deformation with Pseudo Material Effects

Jin Huang, Hongxin Zhang, Xiaohan Shi, Xinguo Liu* and Hujun Bao

State Key Lab of CAD&CG, Zhejiang University, China

{hj, zhx, shixiaohan, xgliu, bao}@cad.zju.edu.cn

Abstract

This paper presents a novel interactive mesh deformation method that can achieve various dynamic material effects, including elastic membrane and cloth effects. In our framework, a mesh is encoded by some differential quantities based on edge length and dihedral angle; and the deformation is formulated as a least square problem for preserving the edge length and dihedral angle via the differential quantities. In order to obtain anisotropic material effects, we further propose an edge weighting scheme based on a user specified vector field. To avoid specifying the local transformations, we set up an iterative scheme for solving the deformation. At last, several examples are presented to show that our approach can interactively generate visually pleasing deformations.

Keywords: Mesh deformation, Gradient Domain Methods, Material Simulation

*Correspondence author

Introduction

The gradient domain methods [1, 2, 3] have shown to be efficient for preserving surface details in mesh deformation and interpolation problems. Gradient domain methods capture local shape information of the meshes by calculating the differential quantities. Because of the ability of building an elegant connection between local detail information and the global shape, gradient domain methods are further enhanced by several authors [4, 5] more recently. Unfortunately, aforementioned methods only penalize isotropic surface details distortions, and they cannot provide explicit length description. Therefore, they cannot achieve deformations effects beyond membranes deformation, e.g. cloth.

In this paper we present a novel method of material aware mesh deformation. Different from most previous work which measure distortion at vertices, we introduce an edge based measure of distortion instead. The penalty of distortion can be amplified or alleviated according to the edge direction and the user specified direction. Edge based scheme enables the penalty of length stretch, and makes the cloth effect possible. To penalize bending, we introduce quantities which reflect the changing of the dihedral angle of two triangles that share an edge. Combining the above two distortion penalties with different intensity, rich material effects can be generated intuitively. Since our distortion measure is non-linear indeed, we set up an iterative scheme to get deformation results by solving a sequence of linear least square problems. At each iteration step, we explicitly estimate the local configuration based on previous result, then solve a linear equation with constant coefficients to reconstruct a new mesh.

The key contributions of our deformation algorithm are: (i) distortion measures of bending and stretching based on differential coordinates which are defined on edges; and (ii) a fast iterative scheme which enables interactive user manipulation and dynamic material adjustment.

Related Work

Surface deformation is an attractive research topic in computer graphics and computer animation. Typical approaches are to design a geometric procedure which analogs human's perception directly, or to simulate a physical procedure which predicts real world phenomenon accurately.

The earlier *geometric based deformation* techniques can be summarized as a subspace embedding approach, such as free form deformation [6, 7], skeleton subspace deformation [8], or multi-resolution style methods [9, 10]. Global vertex coordinates are encoded in subspace, and deformed results are reconstructed forwardly according to users manipulation. To achieve desired results, user can explicitly deform the subspace to approximate shape. Surface properties, such as the penalty of resisting stretch cannot be specified. Therefore the material effects are very difficult to achieve through such kind of techniques which highly depend on user's talent and experience.

The recent proposed gradient domain methods use the differential coordinate to encode the local shape information, and preserve surface details in lease square sense for deformation [1, 2, 4, 11]. Volumetric details can be also measured and preserved to void self-intersection by constructing volumetric graph Laplacian [5]. The ideas of gradient domain can be also applied to manipulating 2D shapes [12].

Our method is highly inspired by above work, and actually can be viewed as an edge based variant for providing material aware deformations. But different from recent proposed ones [13, 14] which aim at controlling propagation manner of local transformations, our approach tend to find out transformed differential quantities using an iterative optimization.

To our knowledge, there are two gradient domain methods which adopt iterative procedure for mesh deformation in the literature. Sheffer et al. [15] propose a rotation invariant shape representation, called pyramid coordinates. Their method updates vertex coordinates according to a set of angles and lengths related to a vertex and its immediate neighbors. Au et al. [16] utilize the surface orientation of the unknown deformed mesh as the Laplacian differential coordinates for deformation computing. Their iterative procedure, therefore, update face related quantities. However, both vertex and face based methods mainly capture geometric features related to surface curvature, hence mesh is always deformed in an elastic membrane style.

Physical based deformation methods are proposed to deform objects according to physical laws. Pioneer work of this area is carried out by Terzoupols and his co-workers [17]. Later, many methods for both off-line and interactive simulation of deformable objects are proposed, such as employing the boundary element method [18] or the finite element method [19]. In general, the whole simulation procedure is driven by a specified deformation energy, and is characterized by an integral equation varying in time. It is important to choose specific methods to solve the integral equation and the corresponding time step, and other related parameters (e.g. damping coefficients). Mass-spring system is one of the simplest computation model in physical based mesh deformation, especially in many cloth effects simulation algorithms such as [20]. In [21], a complicated physical model is described for fast clothing simulation, and adopts implicit integration to provide large time steps. See [22] for a detailed survey on the physically based models.

Unfortunately, the traditional treatments of physical based methods require complex computational machinery thus prohibitive implementation cost. Though the data driven approach [23] can provide efficient simulation performance, it requires a huge amount of precomputation and storage cost. Discrete shells are rapidly adopted by the computer graphics community for their visually convincing results [24]. Bridson et al. [25] propose edge based quantities to simulate cloth dynamics. Both methods measure the bending of triangle mesh surface by using rigorous dihedral angle definition, while our method utilizes a novel differential quantity defined on edge to indicate the magnitude of bending. Therefore our method leads to a simpler energy function for calculation.

Different from physical based formulation, our approach starts from geometric intuition for surface editing applications, and simplifies the physical integral procedure into an iteration scheme. Hence there is no mechanism of time-stamp in our approach.

Distortion Measures

In gradient domain methods, distortion measures are designed to capture shape deformations of a given mesh. According to this principle, local geometric quantities, such as angles and lengths uniquely related to a topological primitive (include vertex, edge or triangle) and its immediate neighbors, should be well preserved.

Vertex-based distortion energy

We first concisely review previous vertex-based approaches. Let $\mathcal{M} = (V, K)$ be a triangle mesh, where $V = \{\mathbf{v}_i \in R^3 | 1 \leq i \leq n\}$ describes the geometric positions of the vertices, and K denotes the topological connectivity. We call the index set $N_i = \{j | (i, j) \in K\}$ to be the 1-ring neighborhood of vertex i. The *differential coordinates* of vertex i can be formulated as:

$$\delta_i = \mathbf{v}_i - \sum_{j \in N_i} w_{ij} \mathbf{v}_j = \mathcal{L}_i(V) \tag{1}$$

where w_{ij} can be either nonuniform [1] or uniform [2] weights, and $\mathcal{L}_i$ is called the *Laplacian operator* at vertex i. The total local distortion measure for a deformation procedure is given by a quadric energy term

$$E = \sum_{i \in K} \|\delta_i' - \mathcal{L}_i(V')\|^2 \tag{2}$$

In Eq. (1), the deformed differential coordinates δ_i' are computed via $\delta_i' = T_i \delta_i$ [1, 2, 5], where T_i indicates the local transformation at vertex i. With a set of constraints $\mathcal{C}(V')$ over mesh $\mathcal{M}$, new deformed vertex positions V' can be obtained by the following least square minimization:

$$\min_{V'} E + \mathcal{C}(V') \tag{3}$$

Commonly, $\mathcal{C}(V')$ indicates soft position constraints at specific vertices $j \in C$, i.e., $\mathcal{C}(V') = \sum_{j \in C} \|\mathbf{v}'_j - \hat{\mathbf{v}}_j\|^2$ with $\hat{\mathbf{v}}_j$ indicating constrained position.

Mesh details, in terms of the difference between center vertex and a linear combination of its neighbors, are preserved by Eq.(3) during a deformation procedure. However, there is no explicit edge length preserving strategy in previous methods. Hence they can only generate elastic membrane effect, and they are difficult to achieve deformation style like cloth. Moreover, previous vertex-based methods only provide isotropic distortion measure, and are difficult to enforce anisotropic distortion penalty.

Edge-based distortion energy

Instead of encoding local information at vertices, we encode differential quantities at edges $\{e_{ij} | (i,j) \in K\}$. Given an edge e_{ij}, let vertex l and m be the third vertex in the two triangles that share e_{ij} respectively (see Fig. 1). We then compute the normal of edge e_{ij} by

$$\mathbf{n}_{ij} = \frac{A_l \mathbf{n}_l + A_m \mathbf{n}_m}{\|A_l \mathbf{n}_l + A_m \mathbf{n}_m\|}, \tag{4}$$

where A_l, A_m and $\mathbf{n}_l, \mathbf{n}_m$ are the areas and the normals of the two adjacent triangles respectively. Finally, we have Laplacian-like quantities along the edge e_{ij}:

$$\left\{\begin{array}{lllll} \rho_{ij} & = & \mathbf{v}_i - \mathbf{v}_j & = & \mathcal{L}^{\rho}_{ij}(V), \\ \sigma_{ij} & = & \mathbf{p}_{ij} - \mathbf{q}_{ij} & = & \mathcal{L}^{\sigma}_{ij}(V), \end{array}\right. \tag{5}$$

with $\mathbf{p}_{ij} = \alpha\mathbf{v}_i + (1-\alpha)\mathbf{v}_j$ and $\mathbf{q}_{ij} = \beta\mathbf{v}_l + (1-\beta)\mathbf{v}_m$. The coefficients α and β are chosen to make σ_{ij} parallel to the edge normal $\mathbf{n}_{ij}$. In Eq. (5), ρ_{ij} is designed to control the edge length, and the magnitude of σ_{ij} is related to the dihedral angle of two adjacent triangles.

Similar to vertex based methods, we have the following two energy terms to measure total local distortions, i.e., local stretch and bending respectively:

$$\left\{\begin{array}{lll} E_{\rho} & = & \sum_{(i,j)\in K} \left\|\rho'_{ij} - \mathcal{L}^{\rho}_{ij}(V')\right\|^2, \\ E_{\sigma} & = & \sum_{(i,j)\in K} \left\|\sigma'_{ij} - \mathcal{L}^{\sigma}_{ij}(V')\right\|^2. \end{array}\right. \tag{6}$$

Here ρ'_{ij} and σ'_{ij} are transformed from the original ones by T_{ij}, i.e., $\rho'_{ij} = T_{ij}\rho_{ij}$ and $\sigma'_{ij} = T_{ij}\sigma_{ij}$. To generate a deformed mesh $\mathcal{M}' = (V', K)$, we come to solve the following optimization problem in least square sense:

$$\min_{V'} \lambda E_{\rho} + \mu E_{\sigma} + \mathcal{C}(V'). \tag{7}$$

In the above equation, λ and μ are two weights for length preserving and bending penalty respectively.

Potential energy

Classical geometric based methods have no sense to express global force controls when manipulating surfaces. However, it is very easy to extend our method for global force controls. Especially, to mimic gravity force which is related to the height of the object, an optional potential energy term can be added into Eq.(7)

$$E_g = \sum_i < \mathbf{g}, \mathbf{v}_i >, \tag{8}$$

where g is a vector representing the direction and the magnitude of gravity. We finally have

$$\min_{V'} \lambda E_\rho + \mu E_\sigma + E_g + \mathcal{C}(V'). \tag{9}$$

Iterative Deformation Procedure

In this section, we will show a general framework for mesh deformation based on the formulation above. We first present an iterative procedure which tends to predict correct Laplacian-like quantities, and solve the mesh in the weighted least square sense. Then we describe how to tailor our framework to be suitable for providing physical plausible effects.

Algorithm

Basically, Eq. (3) and (9) are ill-posed since both T_{ij} and $\mathbf{v}'_i$ are unknown. Direct translation between the local transformations and the global coordinates involves non-linear rotation computation. Therefore, predicting appropriate local transformations plays an important role in gradient domain methods. Most of the previous ones calculate the local transformation for each vertex from handles whose transformations are given by users. In literature, local transformation propagations [1, 5, 26], linear rotation approximation [2], and two passes updating [4] are proposed to build smooth distribution of local transformation over surfaces.

Alternatively, we try to avoid explicitly computing the local transformation T_{ij}. Based on this motivation, we design an iterative algorithm to predict transformed differential quantities. While the surface details are well preserved because of the total distortion measure, the desirable global shapes are ensured by fairly transformed quantity predicting.

Our iterative deformation is an E-M style algorithm. That is, given an initial configuration and constraints, we iteratively update implicit parameters and explicit vertex positions, until a termination condition is satisfied. Precisely, the two sub-steps at the k-th iteration are:

Estimation step. Predict desired edge-based Laplacian quantities by

$$\left\{\begin{array}{lcl} \rho_{ij}^{k+1} & = & \|\rho_{ij}^{0}\|\mathbf{l}_{ij}^{k}, \\ \sigma_{ij}^{k+1} & = & \|\sigma_{ij}^{0}\|\mathbf{n}_{ij}^{k}, \end{array}\right. \tag{10}$$

with $\mathbf{l}_{ij}^{k} = (\mathbf{v}_i^k - \mathbf{v}_j^k)/\|\mathbf{v}_i^k - \mathbf{v}_j^k\|$. Here and in the following, upper subscripts k and $k+1$ represent the numbers of iteration.

Measure step. We update vertex positions by minimizing the following distortion measure

$$\min_{V^{k+1}} \lambda E_{\rho}^{k+1} + \mu E_{\sigma}^{k+1} + E_{g}^{k+1} + \mathcal{C}(V^{k+1}). \tag{11}$$

Here E_ρ^{k+1} and E_σ^{k+1} are simply quadric functions in V^{k+1}

$$\begin{cases} E_\rho^{k+1} & = & \sum_{(i,j)\in K} \left\| \rho_{ij}^{k+1} - \mathcal{L}_{ij}^{\rho}(V^{k+1}) \right\|^2, \\ E_\sigma^{k+1} & = & \sum_{(i,j)\in K} \left\| \sigma_{ij}^{k+1} - \mathcal{L}_{ij}^{\sigma}(V^{k+1}) \right\|^2. \end{cases} \tag{12}$$

Hence solving the above quadratic minimization problem (i.e. Eq. (11)) results in a sparse linear system $A^{k+1}V^{k+1} = b^{k+1}$.

It is worth noting that the Hesse-Matrix (the second partial derivatives) of E_ρ and E_σ are constant. Because the Hesse-Matrix of E_g is always zero, the potential energy contributes to the linear equations on the right side b^{k+1} only. In most cases, we fix the position constraints, i.e., $\mathcal{C}(V^{k+1}) \equiv \mathcal{C}(V^0)$. It follows that the coefficient matrix A^{k+1} of the linear equation is fixed during the iteration. Thus the LU matrix decomposition can be carried out once as a pre-computing step to achieve interactive frame rate (see Table 1).

Pseudo material effects

Although our deformation framework is purely a geometric one, it maintains strong physical intuition. First of all, the iterative update procedure can be an analogue to those real world deformation. After specific parts of a deformable object are dragged to new positions, the deformation is gradually propagated to their neighborhoods within several iteration steps. And the final result is obtained when the deformed object becomes static.

Another important observation is that the two global coefficients λ and μ in Eq. (7) can stand for pseudo material properties. By combining length and dihedral angle penalty with different λ and μ, user can specify various deformation behavior to the triangle mesh. Larger λ can enhance the performance of length preserving. Hence we can provide cloth effects rather than elastic membrane deformation. The coefficient μ stands for stiffness which indicates how hard to bend an object. With smaller λ and larger μ, dihedral angle penalty dominates the deformation results which are similar to elastic thin shells.

Notice that the least square minimization formulation Eq. (7) is made up of quadric terms defined on edges. To more precisely emphasize or alleviate the penalty of distortion, one practical way is to directly weight each term by a scalar value to change its proportion in Eq. (7). In other words, we can formulate a deformation procedure in a weighted least square sense by modifying Eq. (7).

Anisotropic deformation property is very useful for some special fabric with anisotropic structure. Since our distortion measures are edge based, the anisotropic effect can be easily achieved as follows. Given a vector field $\vec{f}(V)$ over mesh $\mathcal{M}$, the intensity of the distortion at edge e_{ij} can be weighted by

$$\kappa_{ij} = \frac{\left|\left\langle \vec{f}_i + \vec{f}_j, \mathbf{v}_i - \mathbf{v}_j \right\rangle\right|}{\left\|\vec{f}_i + \vec{f}_j\right\| \cdot \|\mathbf{v}_i - \mathbf{v}_j\|}. \tag{13}$$

In our implementation we adopt the method proposed in [27] to generate vector fields. Once κ_{ij} has been computed, we revise the total distortion measures as:

$$\begin{cases} E_{\rho}^{k+1} &= \sum_{(i,j)\in K} \kappa_{ij} \left\| \rho_{ij}^{k+1} - \mathcal{L}_{ij}^{\rho}(V^{k+1}) \right\|^2, \\ E_{\sigma}^{k+1} &= \sum_{(i,j)\in K} \kappa_{ij} \left\| \sigma_{ij}^{k+1} - \mathcal{L}_{ij}^{\sigma}(V^{k+1}) \right\|^2. \end{cases} \tag{14}$$

Dynamical adjustment strategy

Under some circumstance, designers may want to adjust pseudo materials dynamically for intervening in the deformation procedure. However, according to the above algorithm description, the coefficient matrix will be updated after tuning related coefficients. Therefore, LU decomposition have to be performed again. Although the back substitution is very fast, the cost of LU decomposition is still quite expensive for models with more than 10K vertices.

To avoid this inconvenience, an alternative way can be applied to varying the material properties at runtime. The key observation is that there is no penalization if we set the penalty term to zero in Eq. (12) at the beginning of each step. It is equivalent to setting $\rho_{ij}^{k+1} = \mathcal{L}_{ij}^{\rho}(V^k)$ or $\sigma_{ij}^{k+1} = \mathcal{L}_{ij}^{\sigma}(V^k)$ explicitly. So parameters ζ and ξ in the following equation can determine the intensity of distortion penalty:

$$\left\{ \begin{array}{lcl} \rho_{ij}^{k+1} & = & (1-\zeta)\mathcal{L}_{ij}^{\rho}(V^k) + \zeta \|\rho_{ij}^0\| \mathbf{l}_{ij}^k, \\ \sigma_{ij}^{k+1} & = & (1-\xi)\mathcal{L}_{ij}^{\sigma}(V^k) + \xi \|\sigma_{ij}^0\| \mathbf{n}_{ij}^k. \end{array} \right. \tag{15}$$

In our experience, it is appropriate to set $\zeta, \xi \in [0,1]$. Modifying parameters ζ and ξ will not affect the coefficient matrix of the sparse linear system. So we can adjust material effects on the fly, which helps users to manipulate deformable objects efficiently.

Model	# vertices	LU (sec)	back-sub (sec)	frame-rate (sec/fr)
Tube	1,410	0.050	0.009	0.030
Plane	10,201	0.510	0.065	0.190
Rabbit	20,001	1.400	0.150	0.270
Octopus	23,898	1.500	0.180	0.300
Armadillo	30,002	4.100	0.400	0.950

Table 1: Statics and timing

Results and Discussion

Figs. 2-6 show several deformation results of various models by our prototype system. Our system enables mesh deformation procedures at an interactive rate. And users can feel like manipulating real objects in it, although our iterative mesh deformation engine is purely geometric based.

Performance statistics are listed in Table 1 for all examples in this paper. In our implementation, we adopt the UMF package as the linear solver. The fifth column of Table 1 indicates the average frame-rate of our interactive system. For each frame, the major computing consumption of our system comprises user interface response, back substitution, normal updating and model displaying. The time consumption of a deformation session depends on two factors: the frame-rate which indicates computation cost of each iteration and the number of total iterations N to reach stable results. N is an important factor but difficult to quantify, for it varies significantly depending on several factors such as the shape of the model, and the locations of the constraints. For all experimented models in this paper, our deformation engine outputs stable results within around 15 iterations. Therefore, users can always obtain satisfying results at an interactive rate.

Fig. 2(a) and 2(b) illustrate deforming a planar mesh by using different combinations of coefficients λ, μ, ζ and ξ. In this example, we fix only 4 corners of the mesh, and drag the central vertex as an additional position constraint for deformation. We set smaller λ or ζ to attain elastic membrane style deformations, while larger λ or ζ result in cloth style deformations. Comparisons of dynamic adjustment coefficients ζ and ξ demonstrated in Fig. 2(b) indicate that they can achieve similar function to λ and μ. When a vector field is assigned (cf. Fig. 2(c)), the anisotropic variant of our deformation algorithm provides desired direction controls.

Fig. 3 and 4 show the results of twist operation. Different from several previous methods [1, 26], we need not to design specific propagations for rotation or other geometric quantities. They are predicted naturally by the iteration procedure. Notice that the surface details or high frequency features can be well preserved in our deformation method.

In Fig. 5, a pseudo cloth deformation example is exhibited by adding the optional gravity term. The consequent sub-images, indicating the iteration procedure, provide strong physical sense. In this example, we only constrain several central points of a square mesh. To demonstrate the power of our method, several other deformation results with their original poses are arranged in Fig. 6. All of them are designed in seconds by a novice user of our prototype system.

Conclusions

In this paper, we presented a novel approach to deforming triangular meshes. Our techniques provide pseudo material effects with interactive performance. In our approach, we aim at using edge based differential quantities to capture the local shape information of meshes. Therefore we can easily penalize bending and preserve edge length when the shape is deformed. Moreover, based on our distortion measure, an iterative scheme is proposed to generate visual pleasant results by solving a sequence of the linear least square problems. During such an iteration procedure, various deformed shape can be created by tuning two global material factors. We achieved anisotropic material effects by weighting the edge distortion penalty with the dot product of the edge and specified direction. In addition, local distortion penalty of each edge can be dynamically adjusted on the fly.

In summary, our techniques provide the users with a simple and intuitive interface to manipulating the mesh objects in physical plausible sense. In a few cases, our deformation algorithm may not converge to a stable state when there are multiple configurations satisfying the user provided constraints. For example, in Fig. 3, specific pose of right handle part may cause unstable results since in real world there are more than one possible deformation ways with identical constraints. We will seek practical strategies to address this issue. It is also interesting to explore non-linear mesh interpolation methods using our edge based differential quantities.

Acknowledgments

The rabbit and Armadillo models are courtesy of Stanford 3D scanning repository. This work is supported in part by the National Basic 973 Program of China (No. 2002CB312102), the National Natural Science Foundation of China (No. 60021201 and No. 60505001), and the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China (No.705027).

References

[1] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[2] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics symposium on Geometry processing*, pages 179–188. Eurographics Association, 2004.

[3] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *SPM ’05*, pages 267–274. ACM Press, 2005.

[4] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24(3):479–487, 2005.

[5] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, 24(3):496–503, 2005.

[6] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86*, pages 151–160. ACM Press, 1986.

[7] Sabine Coquillart. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90*, pages 187–196. ACM Press, 1990.

[8] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00*, pages 165–172. ACM Press, 2000.

[9] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98*, pages 105–114. ACM Press, 1998.

[10] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH '99*, pages 325–334. ACM Press, 1999.

[11] Mario Botsch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, 2004.

[12] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[13] T. Popa, D. Julius, and A. Sheffer. Material aware mesh deformations. In *SMI 2006*, 2006.

[14] Dong Xu, Hongxin Zhang, and Hujun Bao. Non-uniform differential mesh deformation. In *CGI 2006, LNCS 4035*, 2006.

[15] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT04*, 2004.

[16] Oscar Kin-Chung Au, Chiew-Lan Tai, Ligang Liu, and Hongbo Fu. Dual laplacian editing for meshes. *IEEE TVCG*, 12(3):386–395, MAY/JUNE 2006.

[17] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87*, pages 205–214. ACM Press, 1987.

[18] Doug L. James and Dinesh K. Pai. Artdefo: accurate real time deformable objects. In *SIGGRAPH '99*, pages 65–72. ACM Press, 1999.

[19] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01*, pages 31–36, 2001.

[20] Mark Meyer, Gilles Debunne, Mathieu Desbrun, and Alan H. Barr. Interactive animation of cloth-like objects in virtual reality. *Journal of Visualization and Computer Animation (JVCA)*, 2000.

[21] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98*, pages 43–54. ACM Press, 1998.

[22] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boserman, and Mark Carlson. Physically based deformable models in computer graphics. In *Eurographics 2005 state of the art report (STAR)*, 2005.

[23] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *SCA '02*, pages 153–159. ACM Press, 2002.

[24] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *SCA '03*, pages 62–67, 2003.

[25] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03*, pages 28–36, 2003.

[26] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24, pages 601–609. Eurographics, Blackwell, 2005.

[27] Greg Turk. Texture synthesis on surfaces. In *SIGGRAPH '01*, pages 347–354. ACM Press, 2001.

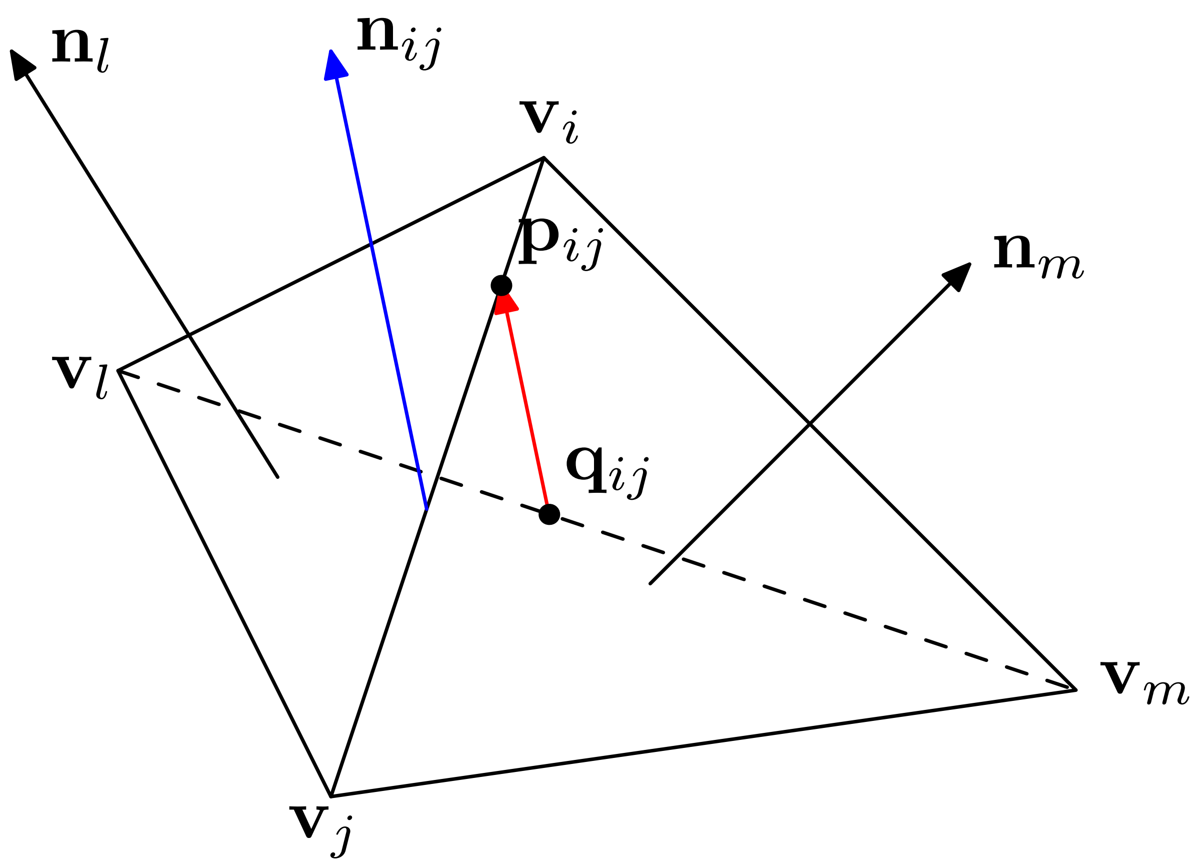
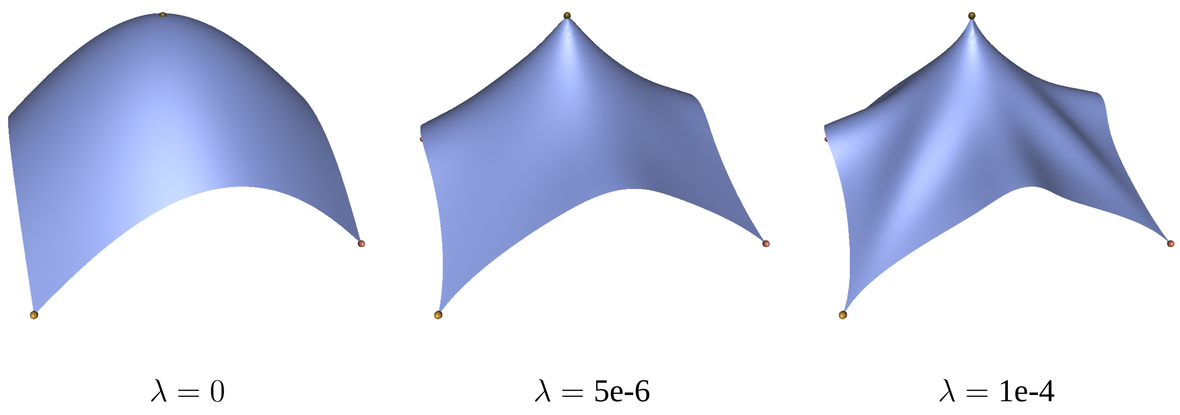
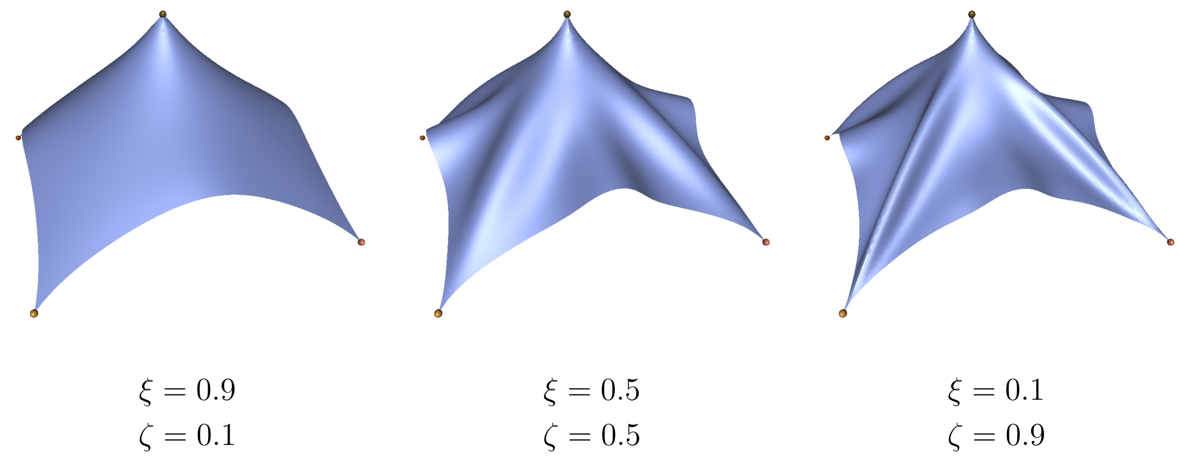


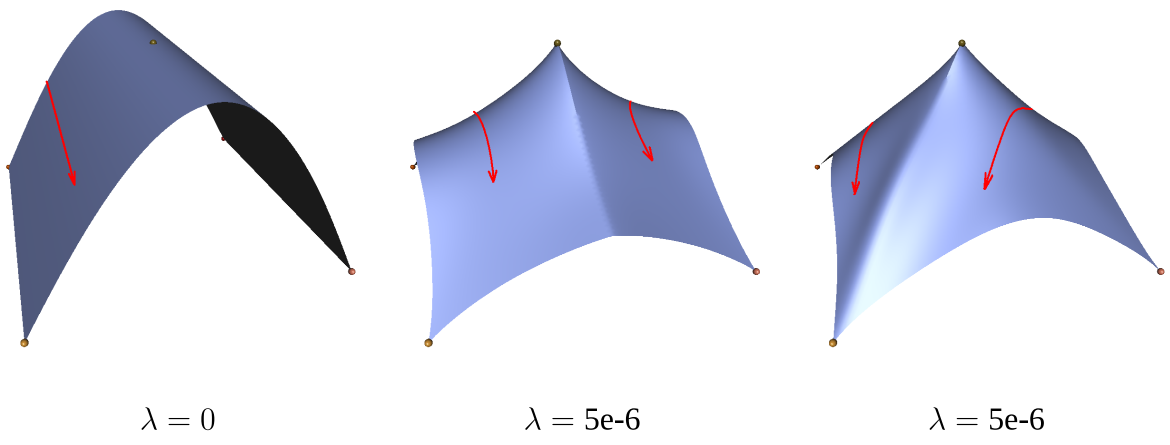
Figure 1: Illustration of computing edge based Laplacian-like quantities.



(a) Effects by λ ($\mu = 1, \zeta = 1, \xi = 1$)



(b) Effects by ζ and ξ($\lambda =$ 1e-4, $\mu = 1$)



(c) Anisotropic effects by vector field ($\mu = 1, \zeta = 1, \xi = 1$)

Figure 2: Illustration of various deformation effects on a planar mesh patch.

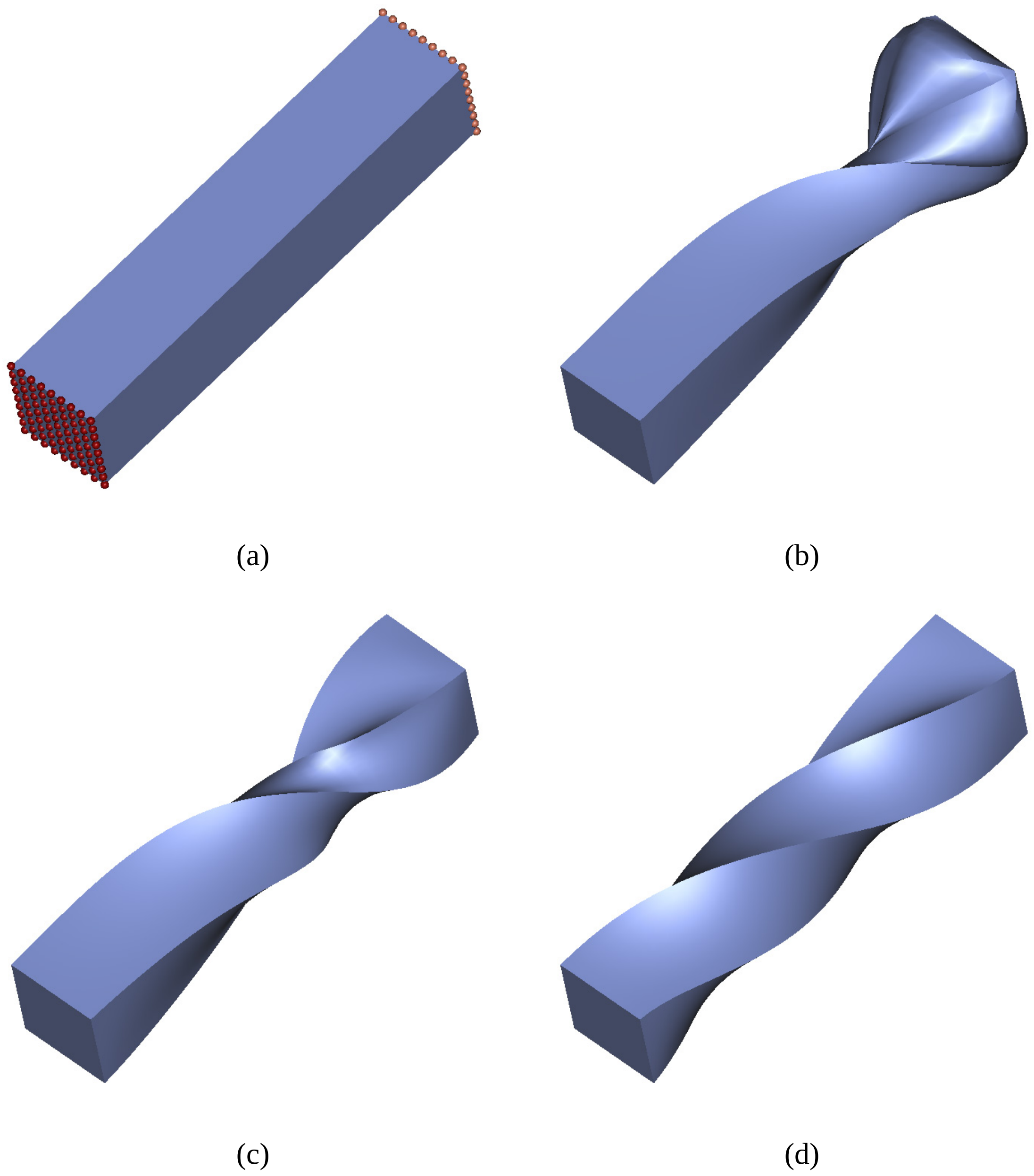


Figure 3: Iterative twisting a simple tube model with $\lambda = 0, \mu = 1, \xi = \zeta = 1$. (a) is the original model. (b)-(d) illustrate results of the first, the 5-th and the 10-th (final) iteration, respectively.

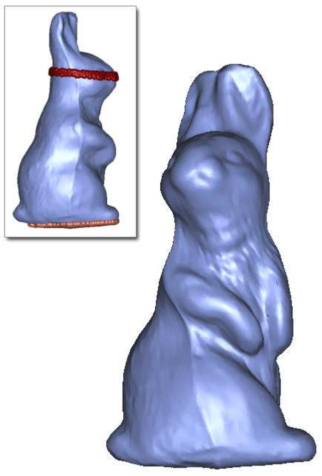


Figure 4: Twisting the head of rabbit so as it can look at itself with $\lambda = 0, \mu = 1, \xi = \zeta = 1$.

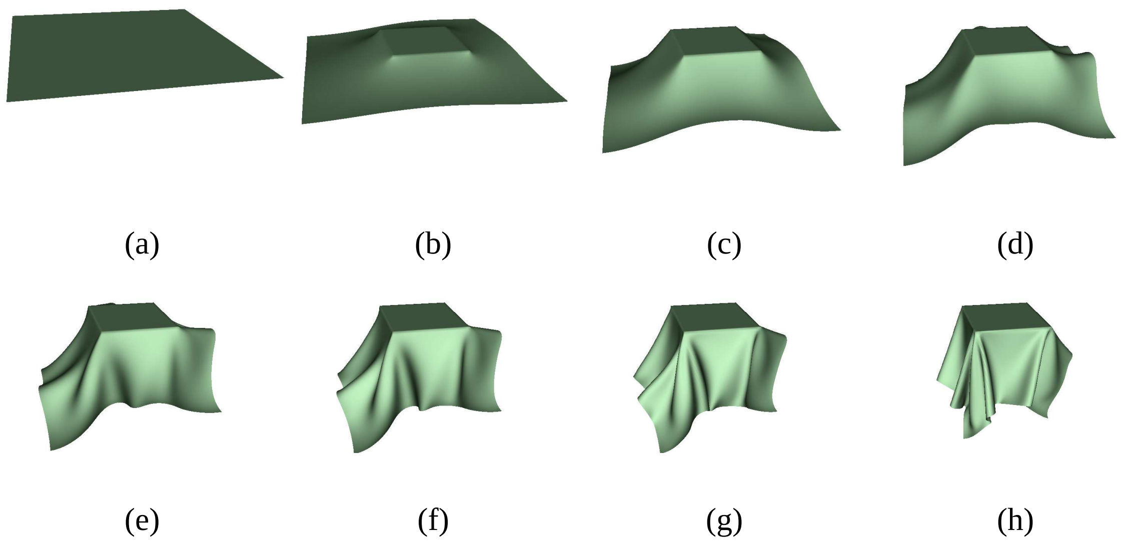
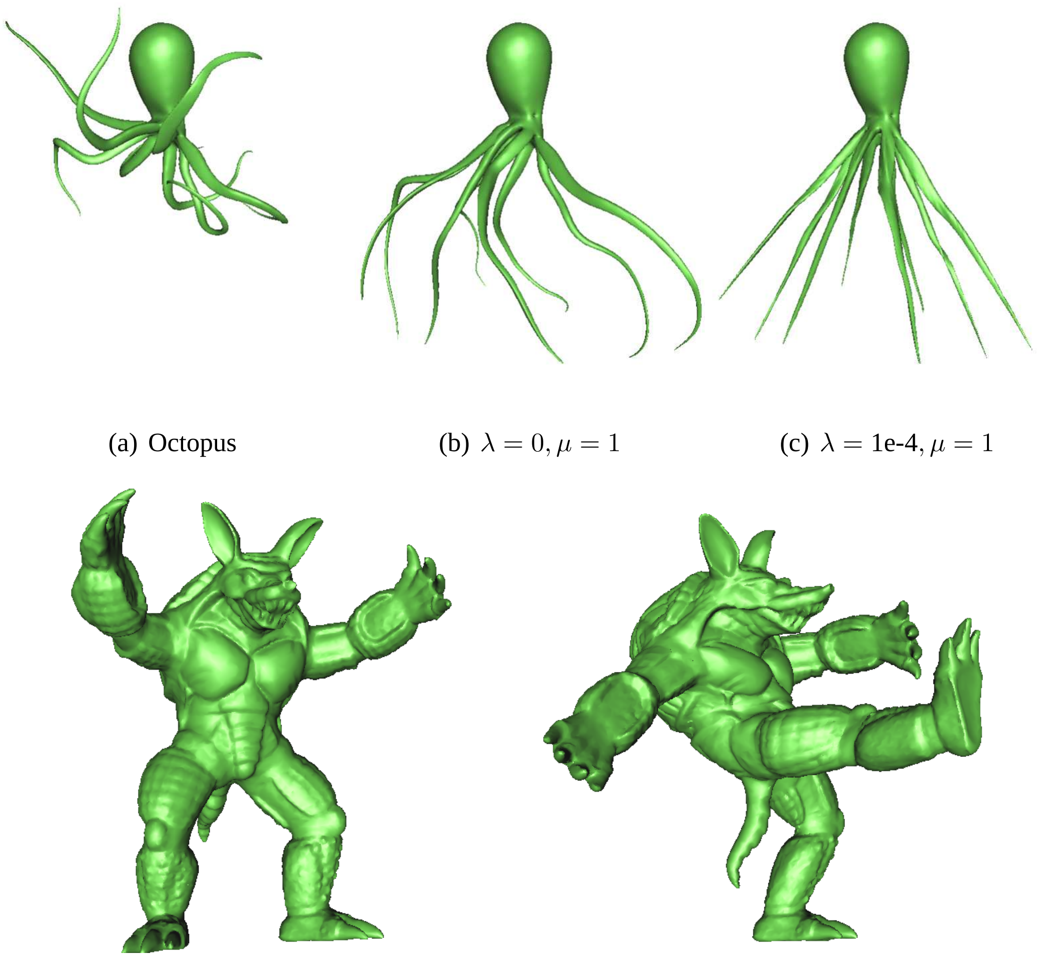


Figure 5: A deformation sequence of pseudo cloth. ($\lambda =$ 1e-3$, \mu = 1, \xi = \zeta = 1$)



(a) Octopus (b) $\lambda = 0, \mu = 1$ (c) $\lambda =$ 1e-4$, \mu = 1$

(d) Armadillo (e) $\lambda = 0, \mu = 1$

Figure 6: Deformation Zoo. In all examples $\xi = \zeta = 1$.