A surface deformation method based on stiffness control

Xuzhou QIN*, Tieru WU* and Yipeng LIU*

* Institute of Mathematics, Jilin University
Changchun, Jilin130000, China
E-mail: wutr@jlu.edu.cn

Received: 10 May 2019 Revised: 14 October 2019 Accepted: 22 January 2020

Abstract
Surface deformation operations commonly require the local details to be preserved as much as possible. In recent years, the As-Rigid-As-Possible (ARAP) shape deformation has gained popularity. Existing ARAP deformation methods define a local cell at each vertex as its 1-ring neighborhoods, and keep the deformation in each local cell as rigid as possible. Since the local cells of adjacent vertices share a single edge, the consistency between adjacent local rigid transformations maintains relative weak. In this paper, we define a new face-based local cell, which expands the overlapping areas between adjacent local cells, thereby enhancing the consistency of adjacent local rigid transformations. Based on this, we further present a stiffness control surface deformation method. The size of the proposed local cell controls global stiffness of the shape. For each local cell, a local stiffness parameter can be specified by the user or can be learned from a set of deforming models automatically to control the local deformation. Through adjusting the size of the local cell and local stiffness parameter, we can generate physically plausible, material-aware deformation results. Based on this local cell, we further propose a shape interpolation method, which has better performance and extrapolation capabilities. Experimental results demonstrate the effectiveness of our method.

Keywords: Local cell, Rigidity, Material aware deformation, Shape interpolation

1. Introduction

Mesh deformation and shape interpolation are useful in variety of applications in computer animations and shape modeling. To animate a 3D mesh, the users commonly use a deformation tool to create new poses or a shape interpolation tool to generate the frames between the poses. Compared with skeleton and cage based deformation methods, surface based deformation techniques are efficient and intuitive to control since it provides a flexible way to directly edit the object by specifying a few position constraints on the geometry of the model. Since most real world objects are made up of different types of materials with different stiffness, the object deformations should exhibit different stiffness behavior. However surface based deformation techniques can't capture the properties of the underlying materials, and thus make it difficult to produce physically plausible deformation results.

In this paper, we introduce a material-aware deformation method, which combines the global and local stiffness parameter to simulate the material properties of the objects. We first introduce a new face-based local cell that enlarges overlapping areas between adjacent local cells, which helps to enhance the consistency of local rigid transformations. When increasing the size $K$ of local cell, the consistency of the local rigid transformations is enhanced, such that the objects are difficult to be stretched and bent (Fig. 2). Therefore, the size $K$ of local cell can be regarded as the global stiffness parameter. We use the size of $K$ to characterize the stiffness of the surface to control the appearance of deformation.

Real-world objects are often exhibited anisotropic stiffness, for articulated shape the joint area tends to be much more flexible, and other areas (e.g., legs and head) tend to be more rigid. To simulate the inhomogeneous object deformation, we assign a local stiffness parameter $w_l$ to each local cell. By adjusting the weight coefficient $w_l$, we can efficiently control the local rigidity of objects (Fig. 3).

To mimic different materials with different stiffness, we design a simple graphical user interface to help users specify
local stiffness parameter $w_l$ for different regions. The user can set the desired local stiffness parameter beforehand and uses a paintbrush tool to assign $w_l$ to surface different regions.

However, manually specifying the local stiffness parameter requires more user efforts to guide the object deformation. In addition to the user specified local stiffness, we propose an automatic method to learn the parameter $w_l$ from a set of deforming shapes which contain implicit knowledge of the material properties. Using the estimated weights, it allows users to generate new deformations which are consistent with the sample set (Fig. 9).

To achieve a physically plausible deformation, we incorporate the global and local stiffness parameter into a nonlinear energy function, and solve the problem using an iterative algorithm. By solving a nonlinear optimization problem, our method produces natural and realistic deformation results. In this work, we concentrate on improving the deformation effects.

Shape morphing or interpolation involves the creation of a natural animation from an initial object to a target pose. It has many applications in computer graphics such as example-driven deformation, or computer animation. In most cases, the geometry of the source and target object is somewhat similar with different poses. In this work, we use the proposed local cell to design a shape interpolation scheme, which has better performance and extrapolation capabilities.

The remainder of the paper is organized as follows: Section 2 reviews the work related to the deformation methods. Section 3 defines a novel face-based local cell. Section 4 gives the detailed algorithm of our deformation framework. Section 5 proposes an automatic method to learn the material properties from a set of deforming shapes. The interpolation algorithm is described in section 6. The experiments and discussions are presented in Section 7. Section 8 presents the conclusion of the paper.

2. Related work

Mesh deformation has been increasing attention in recent years as triangular meshes become a popular representation for 3D models with intricate details. In this section, we briefly review previous work on mesh deformation.

Cage-based deformation algorithms (Ben-Chen et al., 2009, García et al., 2013, Ju et al., 2005, Lipman and Cohen-Or., 2008, Li and Hu., 2013, Zhang et al., 2014) are widely used in commercial software such as 3D Max Studio and Maya. The input object is embedded into a cage, and deformed by manipulating the cage. These algorithms are independent of shape representation and provide the user with a convenient way to deform the shape through manipulating the cage. However, it is a tedious task to construct a compatible cage and it does not consider the object material ingredients. Skinning deformation algorithms (James and Twigg., 2005, Jacobson et al., 2012, Vaillant et al., 2013, Yoshizawa et al., 2007, Yan et al., 2006) construct a skeleton for the input mesh, and then drive the deformation through the skeleton. These algorithms are very intuitive and easy to use. However, it is not a trivial work to automatically construct the skeleton configuration, and the skinning weights in highly deformed regions are difficult to calculate.

Surface based deformation approaches which use intrinsic representations such as local differential coordinates have gained significant popularity. The main idea behind this class of deformation techniques is to minimize a nonlinear energy function representing how well the details are preserved after a deformation. (Fu et al., 2007, Lipman et al., 2004, Sorkine et al., 2004) propose Laplacian coordinates methods. Yu et al. (Yu et al., 2004) manipulate gradient coordinates of the mesh, and then reconstruct the surface from the Poisson equation. These algorithms are linear optimization which suffers from artifacts. (Au et al., 2006, Botsch et al., 2006, Huang et al., 2006, Zhou et al., 2005) propose nonlinear global optimization methods to improve surface deformation results and avoid suboptimal results. Another method to preserve geometric details is to keep local regions rigid. As-rigid-as-possible (ARAP) deformation algorithms introduced by Sorkine and Alexa (Sorkine and Alexa., 2007) define a local cell at each vertex as 1-ring neighborhoods, and ask for the rigidity of local regions and obtain detail-preserving deformations. Because the 1-ring neighborhoods of adjacent vertices share only a single edge, the consistency of neighboring rigid transformations is relatively weak, which leads to flaser during the deformation. Chao et al. (Chao et al., 2010) derive a similar ARAP formulation from the continuous case to achieve volume deformation. Levi and Gotsman (Levi and Gotsman., 2015) improve the ARAP energy with a smooth term. Although various geometry-based algorithms are proposed to improve deformation results, they usually ignore the object material ingredients, and thus make it difficult to generate physically plausible deformations.

Popa et al. (Popa et al., 2006) use a set of affine transformations of local coordinate frames defined per triangle to deform the shape with different material properties. However, they need two stages to compute vertex positions and the user needs to specify the rotational constraints at handles. Our method only needs to specify positional constraints at
handles, which makes the modeling procedure much easier. Yang et al. (Yang et al., 2008) exploit different neighborhood sizes to mimic shape deformations composed of different materials. Their method is based on voxels. The object is embedded into a regular lattice and deformed by manipulating the lattice. However, the high computational costs of the voxelization will increase when the cubic cells are dense and it is difficult to represent deformations at fine scales if the cubic cells are coarse. Our method operates directly on surface, which avoids the complex construction of the lattice. Moreover, we also provide an algorithm to automatically learn the material properties from the given examples without requiring users to specify the rigidity of the object. Chen et al. (Chen et al., 2017) extend ARAP deformation methods from one-ring neighborhoods to larger local neighborhoods to mimic shape deformations composed of different materials. They specify the stiffness of the shape regions by changing the neighborhood size. That is, the smaller neighborhood size is specified for softer areas, whereas the larger neighborhood size for harder areas. However, their method needs high computational costs with the increase of the local cell size. Unlike them, we use local stiffness coefficients within the deformation framework to distribute the deformation according to the local material properties. To simulate objects made of materials with different stiffness, the local stiffness can be specified by the user with an intuitive paint-like interface or can be learnt from a sequence of deforming models.

Shape morphing or interpolation is a fundamental technique widely used in computer graphics. Efficient algorithms for shape interpolation are important for various applications, including animation transfer, morphing, example-driven deformation, and shape exaggeration. Many shape interpolation methods are based on the geometric quantities of the shape. The difference between the different methods lies in the selection of the geometric quantity. Alexa (Alexa., 2003) uses Laplacian coordinates for shape interpolation. Sumner et al., 2005, Xu et al., 2006 use the deformation gradients of the triangles as the geometric quantities followed by Poisson-based fusion. Another example of the geometric quantities uses the linear rotation-invariant coordinates. Lipman et al. (Lipman et al., 2005) propose linear rotation invariant coordinates used for interpolation by combining coordinate frames located at the vertices with connection maps which describe the transformations between neighboring frames. Winkler et al. (Winkler et al., 2010) propose a hierarchical approach that interpolates edge lengths and dihedral angles. Fröhlich and Botsch (Fröhlich and Botsch., 2011) propose a similar algorithm. They add a volume interpolation term to the energy (Winkler et al., 2010), and describe an efficient way to optimize it. Baran et al. (Baran et al., 2009) propose patch-based rotation invariant coordinates and the coordinates record the connection maps between adjacent patches. Levi and Gotsman (Levi and Gotsman., 2015) propose a smooth rotation (SR) enhanced ARAP method for shape interpolation.

3. Face-based local cell

Let $M = \{V,E,F\}$ be a 3D triangular mesh, where $V$ is the set of vertices and $E$ is the set of edges, $F$ is the set of triangles. For each triangle $f_i \in F$, the local cell $C_i$ is defined as 1-ring neighborhoods of its three vertices ($v \in f_i$) (shown in Fig. 1(a)). We call the local cell $C_i$ 1-ring neighborhoods of $f_i$. Figure. 1(b) shows the overlapping areas (red edges) between 1-ring neighborhoods of adjacent triangles, which is larger than single overlapping edge of two adjacent 1-ring vertex neighborhoods (Fig. 1(c)). The local cell can further be extended from 1-ring neighborhoods to $K$-ring neighborhoods by enlarging the respective local neighborhoods of the three vertices. Figure. 1(d) shows the 2-ring neighborhoods of $f_i$. Let $N(f_i;K)$ denotes the set of edges within the $K$-ring neighborhoods of $f_i$. An edge $(i,j) \in N(f_i;K)$ if there exists a vertex $v \in f_i$ such that the vertex $v$ can visit the edge $(i,j)$ through a path no more than $K$ edges (i.e. $(i,j) \in N(v;K)$). Obviously, with increasing $K$, the overlapping areas between adjacent local cells will be enlarged, such that the neighboring local regions will be more closely tied.

![Fig. 1](image)

Fig. 1 (a) Face ($f_i$) 1-ring neighborhoods, (b) the overlapping areas (red edges) between adjacent local cells ($f_n$ and $f_m$), (c) the overlapping edges (red edges) between 1-ring neighborhoods of adjacent vertices ($v_n$ and $v_m$), (d) the 2-ring neighborhoods of triangle $f_i$. 
4. Material-aware deformation

4.1 Deformation energy

To keep the rigid transformations of all triangle K-ring neighborhoods, we define a global energy function as a weighted summation of the deviations of all triangle K-ring neighborhoods transformations from their optimal rigid transformations:

\[ E(\mathbf{v}', \mathbf{R}) = \sum_{i=1}^{n} w_i \sum_{v \in f_i} \frac{1}{d_v} \sum_{(i,j) \in N(v;K)} c_{ij} \| \mathbf{e}_{ij} - \mathbf{R}_l \mathbf{e}_{ij} \|^2 \]  

where \( \mathbf{e}_{ij} = \mathbf{v}_i - \mathbf{v}_j, \) \( \mathbf{e}_{ij} = \mathbf{v}_i - \mathbf{v}_j, \) \( n \) is the number of triangles, \( d_v \) is the degree of vertex \( v. \) \( v \in f_i \) represents the vertex of the triangle \( f_i, \) \( v_i \) is the unknown deformed vertex position, and \( \mathbf{R}_l \) is the rotation matrix to be optimized in each triangle \( K \)-ring neighborhoods. To prevent mesh discretization bias, we use the cotangent weight (Meyer et al., 2003): \( c_{ij} = \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij}), \) where \( \alpha_{ij}, \beta_{ij} \) are the angles opposite to the mesh edge \( (i,j). \)

As demonstrated in (Rivers and James., 2007), the size \( K \) of local cells relates to the surface stiffness. With the increase of neighborhood sizes \( K, \) the consistency of adjacent rigid rotations is enhanced, which makes the material more rigid. To mimic the inhomogeneous object deformation, we incorporate the local stiffness parameter \( w_l \) into the nonlinear energy function Eq. (1). The weight \( w_l \) is a penalty factor for each local cell, which controls the local stiffness of the object. The larger the weight \( w_l \) is, the harder the local cell represents, or in other words, the local cell will look more rigid. However, specifying the local stiffness parameter by manual painting requires more user efforts. We propose a method (Section 5) to automatically learn the weight coefficients from the sample set, such that the learned weights \( w_l \) are suitable for local stiffness.

4.2 Energy function optimization

To minimize the energy function Eq. (1) subjects to the user’s constraints, we employ an alternating minimization strategy to optimize the local rotation matrix \( \mathbf{R}_l \) and the deformed positions \( \mathbf{v}'. \) This is an efficient iterative method, and we now give details for the Eq. (1) optimization.

**Global step.** In this step, the local rotations \( \mathbf{R}_l \) are fixed, and the unknown positions \( \mathbf{v}_i \) can be calculated through Eq. (1) by solving a linear least square problem. Let us compute the partial derivatives w.r.t. \( \mathbf{v}_i':

\[ \frac{\partial E}{\partial \mathbf{v}_i'} = 2 \sum_{i=1}^{n} w_i \sum_{v \in f_i} \frac{1}{d_v} \sum_{(i,j) \in N(v;K)} c_{ij} \left( \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{v}_i'} \right)^T (\mathbf{e}_{ij} - \mathbf{R}_l \mathbf{e}_{ij}) \]  

where:

\[ \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{v}_i'} = \begin{cases} I_{3 \times 3} & i = t \\ -I_{3 \times 3} & j = t \\ 0_{3 \times 3} & \text{else} \end{cases} \]

by setting the gradients to zero \( \frac{\partial E}{\partial \mathbf{v}_i'} = 0, \) we get the following sparse linear system:

\[ \sum_{j \in N(i)} c_{ij} \left( \sum_{v \in f_j} \frac{1}{d(v;K)} \sum_{v \in f_j} w_i \mathbf{e}_{ij} \right) = \sum_{j \in N(i)} c_{ij} \left( \sum_{v \in f_j} \frac{1}{d(v;K)} \sum_{v \in f_j} w_i \mathbf{R}_l \right) \]  

where \( \{v_m : (i,j) \in N(v_m, K)\} \) is a collection of vertices in which the \( K \)-ring neighborhoods of each vertex contain the edge \( (i,j). \) \( \{f_i : v_m \in f_i\} \) is the triangle set containing vertex \( v_m. \) The linear system of Eq. (4) can be rewritten as \( \mathbf{Lx} = \mathbf{b}, \) where \( \mathbf{x} \) is the collection of unknown vertex positions, \( \mathbf{L} \) is semi-definite and can be effectively factorized using Cholesky decomposition. Note that the matrix \( \mathbf{L} \) depends only on the initial local cell configuration, so it is a fixed matrix.
in the iterative optimization process. Thus, in precomputation step, \( L \) only needs to be decomposed once, and solving the linear system amounts to a back substitution. To deform the model by manipulating the constraint vertices on the object, we also need to add the user specified positional constraints \( \mathbf{v}_i = c_i \) into the system Eq. (4).

**Local step.** In local step, the vertex positions \( \mathbf{v}_i \) are fixed, and the local rotation matrix \( \mathbf{R}_i \) is optimized. Following (Sorkine and Alexa., 2007), we get:

\[
\begin{align*}
\argmin_{\mathbf{R}_i \in SO(3)} & \left( \sum_{\mathbf{v} \in \mathcal{V}_i} \frac{1}{d_v} \sum_{(i,j) \in N(v;K)} c_{ij} \| \mathbf{e}_{ij} - \mathbf{R}_i \mathbf{e}_{ij} \|^2 \right) \\
= & \argmin_{\mathbf{R}_i \in SO(3)} \text{tr} \left( -\mathbf{R}_i \sum_{\mathbf{v} \in \mathcal{V}_i} \frac{1}{d_v} \sum_{(i,j) \in N(v;K)} c_{ij} \mathbf{e}_{ij} \mathbf{e}_{ij}^T \right) \\
= & \argmax_{\mathbf{R}_i \in SO(3)} \text{tr}(\mathbf{R}_i \mathbf{S}_i)
\end{align*}
\]

where \( \mathbf{S}_i \) is defined as:

\[
\mathbf{S}_i = \sum_{\mathbf{v} \in \mathcal{V}_i} \frac{1}{d_v} \sum_{(i,j) \in N(v;K)} c_{ij} \mathbf{e}_{ij} \mathbf{e}_{ij}^T
\]

Using singular value decomposition \( \mathbf{S}_i = \mathbf{U}_i \Sigma_i \mathbf{V}_i^T \) leads to the optimal rotation matrix \( \mathbf{R}_i = \mathbf{V}_i \mathbf{U}_i^T \), since \( \text{tr}(\mathbf{R}_i \mathbf{S}_i) = \text{tr}(\mathbf{V}_i \mathbf{U}_i^T \Sigma_i) = \text{tr}(\Sigma_i) \) is the maximum value.

To minimize the Eq. (1), initial values are needed. We set all rotation matrices \( \mathbf{R}_i \) as identity to start the above optimization. Given the user specified positional constraints, we employ the alternating local/global step to iteratively optimize the deformed positions \( \mathbf{v}_i \) and local rotation matrix \( \mathbf{R}_i \). We stop the above iterative process when the energy change \( |\Delta E| = |(E_k - E_{k-1})|/E_k < \varepsilon \), where \( E_k \) is the energy value of the \( k \)th iteration. This stopping criterion works well for all the examples in our experiments.

### 4.3. Discussion about \( K \) and \( w_l \)

As discussion in section 4.1, the size of \( K \) controls the surface rigidity. Figure. 2 shows the deformation results with different size \( K \). When changing the global stiffness \( K \) under the same constraints, the models range from soft material with smaller parameter \( K \) to hard material. When \( K = 1 \), as highlighted in the yellow arrow (Fig. 2(a)), deformation artifacts including self-intersections, unnatural distortions occur in the deformation results. When \( K = 2 \), the distortions are propagated much more uniformly. When further increasing the global stiffness parameter \( K > 2 \) under the same constraints, the shape tends to more rigid so the flexibility is reduced. Table 1 shows the statistics of deformation runtime for the example in Fig.2, where the plane contains 2600 triangles. We note that with the increase of global stiffness parameter \( K \), although the number of iterations is decreasing, the running time for global optimization and local optimization in each iteration is increasing. This is mainly because when increasing neighborhood sizes, the number of non-zero elements in calculating the vector \( \mathbf{b} \) in global optimization and \( \mathbf{S}_i \) in local optimization will increase. Therefore, considering the flexibility, smoothness and interactivity of shape deformation, we fix the global stiffness parameter \( K = 2 \) for all examples in our experiments.

The global stiffness parameter \( K \) controls the whole deformation appearance from soft material to hard material. In general, materials may exhibit anisotropic stiffness. We support such anisotropic behavior by changing local stiffness parameter \( w_l \) across the objects to indicate desired stiffness. Users can select \( w_l \) beforehand, and use paint-like interface to assign \( w_l \) to different areas by painting directly on the surface. In Fig. 3, the middle part of the cuboid is painted green to assign different local stiffness parameter, and other regions which are not painted use default value \( w_l = 1 \). The cuboid is made of two different materials, a softer part and a harder part. As shown in the results, with increasing \( w_l \), the middle part of the cuboid tends to be more rigid. The statistics of deformation runtime for the example (1584 triangles) in Fig.3 are shown in Table 2. In terms of the statistics in Table 2, we find that no matter how much the local stiffness \( w_l \) increases, the running times for global optimization and local optimization in each iteration are similar, i.e., the local
stiffness coefficient $w_l$ does not increase the computational costs for global optimization and local optimization. Therefore, the value of $w_l$ is not constrained in our experiments. The larger the weight $w_l$ is, the more rigid the local material represents.

![Original model (a) (b) (c) (d)]

Fig. 2 A plane is deformed with different global stiffness $K$. (a) $K = 1$, (b) $K = 2$, (c) $K = 3$, (d) $K = 5$.

| $K$–value | Global (ms) | Local (ms) | Iterations | Total time (s) |
|-----------|-------------|------------|------------|----------------|
| $K = 1$   | 6.38        | 6.56       | 275        | 5.1            |
| $K = 2$   | 22.37       | 20.65      | 146        | 9.17           |
| $K = 3$   | 49.81       | 43.84      | 88         | 12.34          |
| $K = 5$   | 116.93      | 103.68     | 47         | 16.86          |

| $w_l$–value | Global (ms) | Local (ms) | Iterations | Total time (s) |
|-------------|-------------|------------|------------|----------------|
| $w_l = 1$   | 13.53       | 13.74      | 204        | 9.36           |
| $w_l = 5$   | 13.20       | 13.50      | 286        | 11.2           |
| $w_l = 10$  | 14.02       | 13.92      | 302        | 11.76          |
| $w_l = 20$  | 13.57       | 14.42      | 353        | 13.21          |

5. Material learning from deforming shapes

Given a set of deformed samples of a model, our goal is to automatically learn the material properties from these deforming shapes. By estimating local stiffness coefficients $w_l$, our method can produce new deformations consistent with the sample set. In our case, since the local stiffness coefficients are derived explicitly, it is convenient for users to modify them if desired.

We assume that a reference model with $n$ triangles and a set of deformed models $q$ ($q = 1, 2, 3 \ldots m$) are available.
To estimate the local deformations, we compute the following 1-ring neighborhood energy value for each of triangles $f_i$:

$$E(f_i, s) = \sum_{v \in f_i} \frac{1}{d_i} \sum_{j \in N(i)} c_{ij} \| e_{ij}^s - R_l e_{ij} \|^2 \quad l = 1, ..., n, s = 1, ..., m,$$

(7)

where $e_{ij}$ and $e_{ij}^s$ are the edge vectors in the reference model and the $s$th deformed model. $R_l$ is the rotation matrix for triangle $f_l$, which can be estimated by the local step in Section 4.2. The amount of deformations the model undergoes locally can be estimated well by choosing the maximum energy value among all samples:

$$\bar{E}(f_i) = \max_s E(f_i, s) \quad (8)$$

Therefore, the above energy value $\bar{E}(f_i)$ can be regarded as the local deformation. The smaller the energy value $\bar{E}(f_i)$ is, the more rigid the local cell represents, and the larger the local stiffness $w_l$ should be, i.e. a large energy value corresponds to a small local stiffness whereas a small energy value for a large local stiffness. To scale the local stiffness in $1$ to $h$, we finally define $w_l$ as follows:

$$w_l = h \cdot \exp \left( - \frac{\bar{E}(f_i)}{\sigma} \right) + 1$$

(9)

where $h$ determines the range of the local stiffness coefficient $w_l$, and $\sigma$ describes the discrete degree of data distribution. We set $h = 7$, $\sigma = 0.0005$ in our experiments.

### 6. Shape Interpolation

In this section, we use the proposed local cell in Section 3 to further introduce a shape interpolation method. Given the source model $M_0$ and target model $M_1$, we denote $C_0^k$ as the 1-ring neighborhoods of the $k$th triangle $f_0^k$ on the source model and $C_1^k$ as the 1-ring neighborhoods of $f_1^k$ on the target model. We first find a linear-transformation $A_k$: $C_0^k \rightarrow C_1^k$ from the source model to the target model by minimizing:

$$E(C_0^k, C_1^k) = \sum_{v \in f_k} \frac{1}{d_i} \sum_{j \in N(i)} c_{ij} \| e_{ij}^k - A_k e_{ij}^0 \|^2 \quad k = 1, ..., n,$$

(10)

setting the derivatives with respect to $A_k$ to zero yields:

$$A_k = \left( \sum_{v \in f_k} \frac{1}{d_i} \sum_{j \in N(i)} c_{ij} (e_{ij}^k)^T \right) \left( \sum_{v \in f_k} \frac{1}{d_i} \sum_{j \in N(i)} c_{ij} (e_{ij}^0)^T \right)^{-1}$$

(11)

Due to the one-to-one correspondence between the linear-transformation $A_k$ and triangle $f_k$, we regard $A_k$ as the linear-transformation of $f_k$ for convenience. Then the linear-transformation $A_k$ can further be decomposed into a rotation part and a stretching part using polar decomposition $A_k = R_k Y_k$. The stretching part is the rotation invariant and can be interpolated directly. We define the interpolation of stretching part between source model $M_0$ and target model $M_1$ at time $t$ as $Y_k = (1 - t)I + t Y_k$, where $I$ is the identity. To be rotation invariant, we define the rotation difference between two adjacent local cells as $dR_{kl} = R_k^T R_l$, where $R_k$ and $R_l$ are the respectively rotational components of the adjacent triangles (i.e. $f_k$ and $f_l$). The rotation difference encodes the relationships between frames. A rotation difference is a rotation matrix that represents a frame in the coordinates of an adjacent frame, which is rotation invariant. We denote the interpolation of the rotation difference between source model and target mode as $\ddot{dR}_{kl} = \text{Slerp}(I, dR_{kl}, t)$, where Slerp is the spherical linear interpolation, which is used for animating 3D rotation in terms of quaternion interpolation.

To solve an intermediate shape $M_t$ at time $t$, we design a reconstruction framework to solve for the vertex positions. According to the definition of the rotation difference, the above linear transformation $A_k$ can be rewritten as: $A_k = R_k dR_{kl} Y_k$, where $R_l$ is the rotation matrix of the triangle $f_l$ adjacent to $f_k$. Therefore, equation (10) can be
rewritten as follows:

\[
E(f_k) = \sum_{v \in f_k} \frac{1}{d_i} \sum_{v' \in \text{adj}(f_k)} c_{ij} \| e_{ij}^1 - A_k e_{ij}^0 \|^2 = \sum_{v \in f_k} \frac{1}{d_i} \sum_{v' \in \text{adj}(f_k)} c_{ij} \| e_{ij}^1 - R_k dR_k i Y_k e_{ij}^0 \|^2 \tag{12}
\]

We note that the above Eq. (12) contains the rotation difference \( dR_{kl} \) and the stretching part \( Y_k \). Therefore, given the interpolated rotation differences \( dR_{kl} \) and the stretching part \( Y_k \) at time \( t \), we can reconstruct new vertex positions \( M_t \) by minimizing the following total energy function:

\[
E(M_t) = \sum_{k=1}^n \sum_{f \in \text{adj}(f_k)} \frac{1}{3} \sum_{v \in f} \frac{1}{d_i} \sum_{v' \in \text{adj}(f_k)} c_{ij} \| e_{ij}^1 - R_k^t \overrightarrow{dR_k i Y_k e_{ij}^0} \|^2 \tag{13}
\]

where \( f_l \in \text{adj}(f_k) \) is the adjacent triangle of \( f_k \); \( d_i \) is the degree of the vertex \( v_i \); \( \overrightarrow{Y_i} = (1 - t)I + tY_i \); \( \overrightarrow{dR_k i} = \text{Slerp}(I, dR_{kl}, t) \). The variables in \( E(M_t) \) are the vertex positions \( e_{ij}^1 \) and local rotation \( R_k^t \). The energy form Eq. (13) is similar to that in (Gao et al., 2016) originally used for each mesh vertex. The Eq. (13) prescribes new interpolated edges by interpolating the stretching part and the rotation difference. It can be seen as safe to interpolate the rotation difference with Slerp instead of directly interpolating the rotations, since the rotation differences between adjacent triangles are usually small for shapes with very large rotations. The Eq. (13) is minimized using the same local/global method described in Section 4.2.

Fig. 4 Bar twist interpolation and extrapolation: \( t = -2; -1.5; 0; 0.5; 1; 1.5, 2 \). (a) is the source shape; (b) is the target shape; the interpolated shape (middle) between (a) and (b) is in green, and the extrapolated shapes on both sides of (a) and (b) are in pink.

7. Experimental results and discussion

The method has been tested for 3D models on a laptop with a 2.0 GHz Intel Core i7-4510U CPU. In this work, we use various datasets from the existing research, including cats, horses (Sumner and Popović, 2004). There is a parameter that is used in our method: the threshold \( \varepsilon \). In all experiments with our implementation, we set \( \varepsilon = 10^{-3} \), which performs well for most of the cases. We use orange arrow and rectangles to show the local deformation effects. Next, we will show more deformation results and demonstrate the effectiveness of our method.

7.1 Results with varying local stiffness coefficients \( w_l \)

In Fig. 5, we compare the deformations with the previous state-of-the-art stiffness control methods (chen et al., 2017). In our algorithm, we use a paintbrush tool to specify local stiffness coefficients \( w_l \) in different surface regions, and other areas which are not painted use default value \( w_l = 1 \). In the method of (chen et al., 2017), we use a paintbrush tool to assign different vertex neighborhood size \( r \) to different regions of the surface, while the default value \( r = 1 \) is used for other areas which are not painted. We fix one end of the bar and move the other end to the same positions. As shown in the results, with the increase of vertex neighborhood size \( r \) in the painted areas (the second row), the unpainted regions (\( r = 1 \)) are affected by the painted regions, which makes the unpainted regions difficult to bend so the flexility is reduced. The anisotropic materials disappear in the deformation results of (chen et al., 2017) when the neighborhood size \( r \) increases, which looks unreal. Our method produces reasonable results without such artifacts by varying local...
stiffness coefficients $w_l$ rather than the local neighborhood size $r$. With increasing $w_l$, the painted areas tend to be more rigid, and the flexibility of the unpainted regions ($w_l = 1$) remains unchanged (the first row).

The deformation results generated by our method can also be compared with those generated by the existing linear or nonlinear methods. In Figs. 6-7, the user specifies some rigid regions for articulated models by a paintbrush. The joint areas should be much more flexible for articulated models. By specifying the local rigid regions, state-of-the-art rigidity-control method (chen et al., 2017) and our algorithm produce much more realistic and natural results than existing methods (Sorkine et al., 2004, Sorkine and Alexa., 2007, Levi and Gotsman., 2015). Existing rigidity-control method (chen et al., 2017), however, reduces the flexibility in joint areas since the unpainted regions are affected by the painted regions with large neighborhood size $r$, and thus looks not real in joint areas (Fig. 6(d)). We also use numerical ways to explain the deformation results. We calculated the distortion of each vertex 1-ring neighborhood, according to the ARAP energy in (Sorkine and Alexa., 2007). The energy values can reflect the local distortion distribution. The smaller the energy value is, the more rigid the local shape represent. A reasonable deformation should have a large distortion at the joint part and small at the non-joint part. It is not difficult to see from Table.3 and Table.4 that the max and average energy values of our algorithm at non-joint part are less than other three algorithms, and opposite at joint part. Figs.6 (a)-(d) intuitively draw the energy values for each vertex, red colors are associated with larger energy values and green colors are associated with smaller energy values. (Sorkine et al., 2004, Sorkine and Alexa., 2007) produces distortion or collapse artifacts at the joints under large-scale translation and (Levi and Gotsman., 2015) looks like soft plastic without joints. Our method produces reasonable results without artifacts.

![Fig. 5 Comparison of deformation results. 1. Our results with different local stiffness. 1(a): $w_l = 5$, 1(b): $w_l = 15$, 1(c): $w_l = 30$, 2. results of (chen et al, 2017) with different vertex neighborhood size. 2(a): $r = 3$, 2(b): $r = 6$, 2(c): $r = 9$.](image-url)
7.2 Results with automatically learned local stiffness coefficients \( w_l \)

We also test our method for estimating the local stiffness coefficients from a collection of shapes with the same connectivity. Figures 8, 9 and 10 show our results. In Fig. 8, we use two sample poses (Fig. 8 (b)) to demonstrate the correctness of our proposed algorithm. The automatically learned local stiffness coefficients \( w_l \) are shown using color.

Table 3. Statistics of the energy value on the legs.

|                  | (Sorkine and Alexa., 2007) | (Levi and Gotsman., 2015) | (chen et al., 2017) | Our method |
|------------------|-----------------------------|---------------------------|---------------------|------------|
| Max value        | 0.1272                      | 0.1904                    | 0.1194              | 0.0854     |
| Mean value       | 0.0558                      | 0.0767                    | 0.054               | 0.0462     |

Table 4. Statistics of the energy value at the joints.

|                  | (Sorkine and Alexa., 2007) | (Levi and Gotsman., 2015) | (chen et al., 2017) | Our method |
|------------------|-----------------------------|---------------------------|---------------------|------------|
| Max value        | 0.2953                      | 0.3172                    | 0.5682              | 0.8340     |
| Mean value       | 0.1067                      | 0.0683                    | 0.0668              | 0.1246     |

Fig. 6 Comparison of deformation results, (a) ARAP deformation result (Sorkine and Alexa., 2007), (b) deformation result (Levi and Gotsman., 2015), (c) anisotropic material specified by user; (d) deformation result (chen et al., 2017) with mixed \( r \)-ring neighborhoods (\( r = 8 \)), (e) our result with local stiffness coefficients (\( w_l = 12 \)).

Fig. 7 Comparison of deformation results, (a) The deformation results by (Sorkine et al., 2004), (b) deformation result (Levi and Gotsman., 2015), (c) the local stiffness (\( w_l = 12 \)) specified by user, (d) our method with the specified local stiffness.
map in Fig. 8(a). With the increasing $w$, the color gradually changes from green to red. Our method can effectively identify the rigid parts and local non-rigid regions by learning the local stiffness $w$ from two deforming shapes.

In Fig. 9, we use 48 sample poses to automatically estimate the local stiffness. As expected, the learned local stiffness coefficients $w$ are correct (Fig. 9(a)), showing more flexible regions at the joints and stiffer regions along the bones and head, similar to the estimated materials in (chen et al., 2017). Since the method of (chen et al., 2017) estimates material based on each vertex, it requires that the vertices between the original pose and the deformed poses must be processed one-to-one. As shown in Fig. 9(d), our method yields realistic-looking deformation results which are consistent with the sample set. However, existing deformation methods (Sorkine and Alexa., 2007, Levi and Gotsman., 2015) can’t preserve the rigid parts and occur distortions around joints.

Figure 10 shows an example of automatically estimating the local stiffness coefficients by using 10 sample poses. The deformation results of (Sorkine and Alexa., 2007) produce collapse artifacts at the joints and the deformation results of (Levi and Gotsman., 2015) smooth out the joints. Our method produces natural and reasonable results by automatically learning the material properties from the sample set.
7.3 Results with interpolation and extrapolation

Figures 4, 11, 12, 13 show our interpolation and extrapolation results. Figure 4 shows an example of blending the bar models. It can be shown that the interpolation and extrapolation results are correctly produced. By changing the parameter \( t \), our method can effectively interpolate and extrapolate them. The weights used for the source and target models are \( t \) and \( 1 - t \). Figure 11 gives the morphing results of a stick and a spiral. Due to the large rotations, it is a challenging case for interpolation. As shown in results, the interpolation and extrapolation results are correctly produced even for large deformations. The poses of the spiral and the bar in Figs. 4 and 11 are similar as in (Levi and Gotsman., 2015). The difference between the results of the different methods is not obvious. Since the interpolation method in (Levi and Gotsman., 2015) computes the maps’ differentials based on each triangle, it requires that the triangles between the source model and target model be handled one-to-one. In Fig. 12, we show the man interpolation/extrapolation sequence. The morphing results generated by our algorithm are natural and reasonable. Figure. 13 shows an example of interpolating the lion models with different parameters \( t \in (0,1) \). Table.5 shows the total number of polygons and total deformation or interpolation time for some examples.

Fig. 10 Comparison of deformation results. (a) Estimated local stiffness. The color map illustrates the change of local stiffness from \( w_l = 1 \) to \( w_l = 8 \). (b) ARAP deformation result (Sorkine and Alexa., 2007). (c) deformation result (Levi and Gotsman., 2015). (d) our result using the learned local stiffness coefficients \( w_l \).

Fig. 11 Spiral interpolation and extrapolation: \( t = 1.25; 1.0; 0.8; 0.6; 0.4; 0.2; 0; -0.25; -0.5; -0.75 \). Source and target shapes are in brown, interpolated shapes are in green, and extrapolated shapes are in pink.
In this paper, we propose a new face-based local cell. Based on this, we present a shape deformation method that incorporates the size of local cell and local stiffness coefficients into a geometric deformation framework. By adjusting the local stiffness parameters suitable to the material, our deformation framework yields natural and physically plausible deformations. The material properties can be specified by users with a simple paintbrush interface, or learned from a set of given poses. Finally, we use the proposed local rigid cell to propose a shape interpolation method, which works effectively in both interpolation and extrapolation.

Although our method can handle a large variety of shapes with fine details, it is challenging to deal with the poor triangle quality (containing slim triangles). When the triangle size differs greatly from part to part, our method will produce suboptimal results. Therefore, in our deformation framework, the size of triangles should be as regular as possible. Another limitation is that currently the global stiffness $K$ still needs to be specified by the users to simulate the rigidity of the object. For future work we would like to design a method to automatically decide the global stiffness $K$ that mimics specified material in the real world, (e.g. wood, iron or steel, not only restricted to joint models).

**Table 5.** Performance data measured in seconds. The first two models are the result of a shape deformation, and the last three models are the result of a shape interpolation at time $t = 0.5$.

| Original model | Vertices | Triangles | Iterations | Total time (s) |
|----------------|----------|-----------|------------|---------------|
| Horse          | 8431     | 16843     | 161        | 11.623        |
| Cat            | 7207     | 14410     | 306        | 18.577        |
| Spiral         | 1034     | 2064      | 531        | 5.87          |
| Lion           | 5000     | 9996      | 93         | 4.97          |
| Man            | 10002    | 20000     | 141        | 16.03         |

8. Conclusion

In this paper, we propose a new face-based local cell. Based on this, we present a shape deformation method that incorporates the size of local cell and local stiffness coefficients into a geometric deformation framework. By adjusting the local stiffness parameters suitable to the material, our deformation framework yields natural and physically plausible deformations. The material properties can be specified by users with a simple paintbrush interface, or learned from a set of given poses. Finally, we use the proposed local rigid cell to propose a shape interpolation method, which works effectively in both interpolation and extrapolation. Although our method can handle a large variety of shapes with fine details, it is challenging to deal with the poor triangle quality (containing slim triangles). When the triangle size differs greatly from part to part, our method will produce suboptimal results. Therefore, in our deformation framework, the size of triangles should be as regular as possible. Another limitation is that currently the global stiffness $K$ still needs to be specified by the users to simulate the rigidity of the object. For future work we would like to design a method to automatically decide the global stiffness $K$ that mimics specified material in the real world, (e.g. wood, iron or steel, not only restricted to joint models).
Acknowledgments

This work was supported by the National Natural Science Foundation of China, No.61373003, and the National Natural Science Foundation of China, No.61872162.

References

Alexa, M., Differential coordinates for local mesh morphing and deformation. The Visual Computer, Vol.19, No.2-3 (2003), pp.105–114.

Au, O. K. C., Tai, C. L., Liu, L. and Fu, H., Dual Laplacian editing for meshes, IEEE Transactions on Visualization and Computer Graphics, Vol.12, No.3 (2006), pp.386–395.

Baran, I., Vlasic, D., Grinspun, E. and Popović, J., Semantic deformation transfer, In ACM Transactions on Graphics (TOG), Vol.28, No.3 (2009), pp.1–6.

Ben–Chen, M., Weber, O. and Gotsman, C., Variational harmonic maps for space deformation, ACM Transactions on Graphics, Vol. 28, No.3 (2009), pp.1–11.

Botsch, M., Pauly, M., Gross, M. and Kobbelt, L., PriMo: coupled prisms for intuitive surface modeling, In Symposium on Geometry Processing (2006), Cagliari, Sardinia, pp. 11–20.

Chao, I., Pinkall, U., Sanan, P. and Schröder, P., A simple geometric model for elastic deformations, ACM transactions on Graphics (TOG), Vol.29, No.4 (2010), pp.157–166.

Chen, S. Y., Gao, L., Lai, Y. K. and Xia, S., Rigidity controllable as-rigid-as-possible shape deformation, Graphical Models, Vol.91, (2017), pp.13–21.

Fröhlich, S. and Botsch, M., Example–driven deformations based on discrete shells, In Computer graphics forum, Vol.30, No.8 (2011), pp.2246–2257.

Fu, H., Au, K. C. and Tai, C. L., Effective Derivation of Similarity Transformations for Implicit Laplacian Mesh Editing, Computer Graphics Forum, Vol.26, No.1 (2007), pp.34–45.

García, F. G., Paradinas, T., Coll, N. and Patow, G., *Cages: A multilevel, multi-cage-based system for mesh deformation, ACM Transactions on Graphics (TOG), Vol.32, No.3 (2013), pp.24:1–24:13.

Gao, L., Lai, Y. K., Liang, D., Chen, S. Y. and Xia, S., Efficient and flexible deformation representation for data-driven surface modeling, ACM Transactions on Graphics (TOG), Vol.35, No.5 (2016), pp.1–17.

Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L. Y., Bao, H. J., Guo, B. N., Teng, S. H. and Shum, H. Y., Subspace gradient domain mesh deformation, In ACM Transactions on Graphics (TOG), Vol.25, No.3 (2006), pp.1126–1134.

James, D. L. and Twigg, C. D., Skinning mesh animations, ACM Transactions on Graphics, Vol.24, No.3 (2005), pp.399–407.

Jacobson, A., Baran, I., Kavan, L. and Sorkine, O., Fast automatic skinning transformations, ACM Transactions on Graphics, Vol.31, No.4 (2012), pp.1–10.

Ju, T., Schaefer, S. and Warren, J., Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics, Vol. 24, No.3 (2005), pp.561–566.

Levi, Z. and Gotsman, C., Smooth rotation enhanced as-rigid-as-possible mesh animation, IEEE transactions on visualization and computer graphics, Vol.21, No.2 (2015), pp.264–277.

Li, X. and Hu, S., Poisson Coordinates, IEEE Transactions on Visualization and Computer Graphics, Vol.19, No.2 (2013), pp.344–352.

Lipman, Y., Levin, D. and Cohen-Or, D., Green Coordinates, ACM Transactions on Graphics, Vol.27, No.3 (2008), pp.1–10.

Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rossi, C. and Seidel, H. P., Differential coordinates for interactive mesh editing, In Proceedings. Shape Modeling International (2004), Genova, Italy, pp.181–190.

Lipman, Y., Sorkine, O., Levin, D. and Cohen-Or, D., Linear rotation-invariant coordinates for meshes, In ACM Transactions on Graphics (TOG), Vol.24, No.3 (2005), pp.479–487.

Meyer, M., Desbrun, M., Schröder, P. and Barr, A. H., Discrete differential-geometry operators for triangulated 2-manifolds, In Visualization and mathematics III (2003), Springer, Berlin, Heidelberg, pp.35–57.

Popa, T., Julius, D. and Sheffer, A., Material-aware mesh deformations, In IEEE International Conference on Shape Modeling and Applications (2006), IEEE Computer Society, pp.141–152.
Rivers, A. R. and James, D. L., FastLSM: fast lattice shape matching for robust real-time deformation, In ACM Transactions on Graphics (TOG), Vol.26, No.3 (2007), pp.82.
Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C. and Seidel, H. P, Laplacian surface editing, In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Nice, France, 2004, pp. 175–184.
Sorkine, O. and Alexa, M., As-rigid-as-possible surface modeling, In Symposium on Geometry processing (2007), Barcelona, Spain, pp. 109–116.
Sumner, R. W., Zwicker, M., Gotsman, C. and Popović, J., Mesh-based inverse kinematics, In ACM transactions on Graphics (TOG), Vol.24, No.3 (2005), pp.488–495.
Sumner, R. W., and Popović, J., Deformation transfer for triangle meshes, ACM Transactions on Graphics (TOG), Vol.23, No.3 (2004), pp.399–405.
Vaillant, R., Barthe, L., Guennebaud, G., Cani, M. P., Rhomer, D., Wyvill, B. and Paulin, M., Implicit skinning: real-time skin deformation with contact modeling, ACM Transactions on Graphics (TOG), Vol.32, No.4 (2013), pp.1–12.
Winkler, T., Drieseberg, J., Alexa, M. and Hormann, K., Multi-scale geometry interpolation. In Computer graphics forum, Vol.29, No.2 (2010), pp.309–318.
Xu, D., Zhang, H., Wang, Q. and Bao, H., Poisson shape interpolation, Graphical models, Vol.68, No.3 (2006), pp.268–281.
Yan, H. B., Hu, S. M. and Martin, M., Skeleton-Based Shape Deformation Using Simplex Transformations, Proceedings of Computer Graphics International Conference (2006), CGI 2006, Hangzhou, China, pp.66–77.
Yang, W., Feng, J. and Jin, X., Shape deformation with tunable stiffness, The Visual Computer, Vol.24, No.7-9 (2008), pp.495–503.
Yoshizawa, S., Belyaev, A. and Seidel, H. P., Skeleton-based Variational Mesh Deformations, Computer Graphics Forum, Vol.26, No.3 (2007), pp.255–264.
Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B. and Shum, H. Y., Mesh editing with poisson-based gradient field manipulation, In ACM Transactions on Graphics (TOG), Vol.23, No.3 (2004), pp.644–651.
Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K. and Liu, L., Local barycentric coordinates, ACM Transactions on Graphics, Vol.33, No.6 (2014), pp.1–12.
Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B. and Shum, H. Y., Large mesh deformation using the volumetric graph laplacian, In ACM transactions on graphics (TOG), Vol.24, No.3 (2005), pp.496–503.