Rigidity Controllable As-Rigid-As-Possible Shape Deformation

Shu-Yu Chen[a,b], Lin Gao[a,*], Yu-Kun Lai[c], Shihong Xia[a]

[a] *Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences*
[b] *University of Chinese Academy of Sciences*
[c] *School of Computer Science and Informatics, Cardiff University, UK*

Abstract

Shape deformation is one of the fundamental techniques in geometric processing. One principle of deformation is to preserve the geometric details while distributing the necessary distortions uniformly. To achieve this, state-of-the-art techniques deform shapes in a locally as-rigid-as-possible (ARAP) manner. Existing ARAP deformation methods optimize rigid transformations in the 1-ring neighborhoods and maintain the consistency between adjacent pairs of rigid transformations by single overlapping edges. In this paper, we make one step further and propose to use larger local neighborhoods to enhance the consistency of adjacent rigid transformations. This is helpful to keep the geometric details better and distribute the distortions more uniformly. Moreover, the size of the expanded local neighborhoods provides an intuitive parameter to adjust physical stiffness. The larger the neighborhood is, the more rigid the material is. Based on these, we propose a novel rigidity controllable mesh deformation method where shape rigidity can be flexibly adjusted. The size of the local neighborhoods can be learned from datasets of deforming objects automatically or specified by the user, and may vary over the surface to simulate shapes composed of mixed materials. Various examples are provided to demonstrate the effectiveness of our method.

Keywords: shape deformation, rigidity, neighborhood, geometric modeling, as rigid as possible

1. Introduction

Shape deformation is a fundamental research area in computer graphics. For character animation, skeleton based methods are widely used [1, 2]. Such methods however need the users to take extra effort to build the skeletons. Alternatively, some deformation methods [3, 4] take cages (simplified geometry enclosing the deforming shapes) as proxies to deform the shapes. Again efforts are needed to build cages.

Compared with skeleton and cage based deformation methods, surface based deformation methods are more intuitive and more flexible to model a variety of shapes,

*Email: gaolin@ict.ac.cn

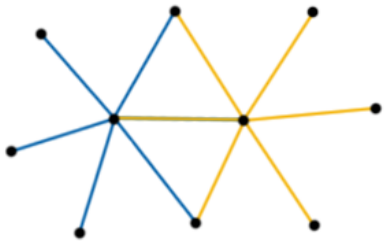


Figure 1: The single overlapping edge of two adjacent 1-ring vertex neighborhoods.

with no need to cope with extra proxies. Laplacian deformation methods [5, 6, 7, 8] have been explored extensively for surface based deformation. The difference between the Laplacian coordinates of the deformed and the original shapes is minimized to keep the local geometric details. However, both the positional and rotational constraints for the deformation handles are required for these methods to work. As shown in [9], positional and rotational constraints need to be assigned *compatibly* to avoid artifacts. This is non-trivial and requires additional effort/expertise from the user.

Another general approach to keep geometric details is to deform shapes locally rigidly, just as rotating and translating shapes globally rigidly would not change their geometry. This principle is modeled as an As-Rigid-As-Possible (ARAP) energy which has been widely used in geometric processing. Based on this energy, Sorkine et al. [10] present a mesh deformation method. Only positional constraints need to be specified at deformation handles. The local rotation of the deformed surface can be estimated automatically during the iterative optimization. This makes interactive modeling much easier and substantially reduces the effort of modeling tasks. The ARAP deformation effectively preserves geometric features and distributes distortions uniformly, which leads to more visually pleasing deformation results than previous methods. The ARAP deformation formulation has also been integrated into various applications in geometry processing. The ARAP deformation method has recently been improved for efficiency [11] and effectiveness [12].

The mechanism of the traditional ARAP deformation [10] is to keep geometric features by deforming the shape locally rigidly. To distribute distortions uniformly over surfaces, local transformation consistency is enforced based on 1-ring neighborhoods of vertices. As shown in Figure 1, the 1-ring neighborhoods of adjacent vertices share a single edge, so the consistency constraint of neighboring rigid transformations is relatively weak. As a result, ARAP deformation results behave as if they are made of soft plastic material.

In this paper, we further explore the ARAP energy. Our key observation is that expanding local neighborhoods will enlarge overlapping areas between adjacent vertices which helps to enhance the coherence of local rigid transformations. As a result, the size of local neighborhoods provides a feasible way to control the appearance of deformation. The larger the size is, the better the local geometric details would be kept, or in other words, the material will look more rigid. By varying this parameter, the appearance of deformation ranges from softer plastic material with smaller

local neighborhoods to harder material such as iron. The former is elastic and flexible, whereas the latter is more rigid and harder to bend/stretch. Another advantage of such enhanced rigid transformation consistency is it reduces the variability of local rigid transformations, and hence the optimization will converge with fewer iterations.

Real-world objects are often composed of different materials. Our approach allows such situations to be well simulated by varying the neighborhood sizes across the objects to indicate desired stiffness. Shapes with different local neighborhood sizes can be effectively optimized in a unified framework.

The contributions of this work are summarized as follows:

- We propose a rigidity controllable deformation method by using ARAP deformation with adjustable neighborhood sizes. By using a local neighborhood size suitable to the material, our method produces more natural deformation than state-of-the-art methods.
- Our unified framework allows varying local neighborhood sizes across the surface, simulating objects made of materials with different stiffness. Realistic deformation results are obtained for such cases.
- In addition to user specified neighborhood sizes, we also develop an automatic method to set neighborhood sizes by analyzing a collection of deforming objects, such that the neighborhood sizes are adapted to local stiffness.

We review the most related work in Section 2. The detailed algorithm is described in Section 3. Results and discussions are presented in Section 4. Finally, limitations and future work are given in Section 5.

2. Related Work

Shape deformation is an active research area in computer graphics with a large amount of related research work. For complete and detailed surveys please refer to [13, 14, 15]. In this section, we review the work most related to ours. In order to simulate realistic shape deformation, the pioneer research work [16, 17] deforms the shapes according to the physical laws. These physically based methods however are computationally intensive and the parameters derived from physical rules cannot be adjusted intuitively.

To generate visually pleasing deformation results, geometric details should be preserved after the shape is deformed. One typical approach is to preserve the Laplacian differential coordinates [5, 6, 7, 8] during the shape deformation. These differential coordinates based methods need the user to specify compatible positional and rotational constraints for deformation handles. As shown in [18], incompatible constraints will introduce artifacts. Popa et al. [19] deform the shape with different material properties based on the deformation gradient method. Again, rotational constraints of the deformation handles should be assigned. Our method allows materials with different stiffness to be simulated, while only requiring positional constraints at handles which makes the modeling procedure much easier. For human body deformations, Murai et al. [20] propose a sophisticated mathematical model to learn parameters for simulating

deformation dynamics of soft human tissues. Compared with this work, our work uses a simpler model and can deal with general shapes.

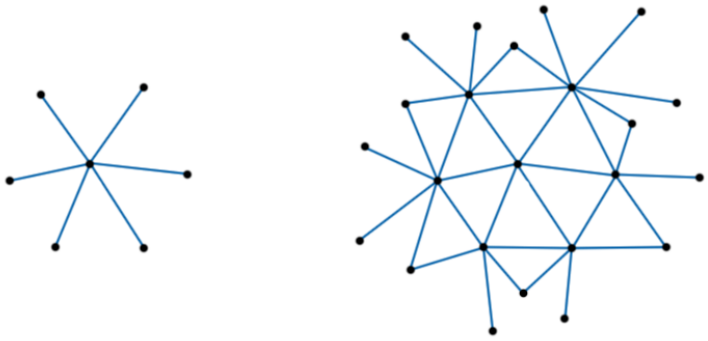
Another approach to preserving geometric details is to keep deformation rigidly. Global rigid transformation while being distortion free is not suitable when non-rigid deformation is involved. Deforming shapes locally rigidly keeps geometric details and makes less distortion. This concept has been modeled as the As-Rigid-As-Possible (ARAP) deformation energy, which has been widely used in geometric modeling, such as shape interpolation [21, 22] and 2D shape manipulation [23]. Sorkine et al. [10] propose a 3D mesh deformation method by using this ARAP technique. This state-of-the-art work often deforms shapes with visually pleasing results. Optimizing the ARAP energy in the L_1 norm instead of the traditional L_2 norm tends to distribute the distortions sparsely to fewer places thus keep geometric features better for most areas [24]. Zohar et al. [12] augment the ARAP energy with a rotation difference term to improve smoothness of relative rigid rotations (SR-ARAP). Gao et al. [25] blend several reference shapes with the ARAP energy for data-driven morphing. The ARAP based shape optimization framework has also been used for shape registration [26] and parametrization [27]. Chao et al. [28] present a continuous ARAP energy formulation. Based on optimizing ARAP energy in the 2-ring neighborhood, Gao et al. [29] propose an approach to data-driven shape deformation. For animation of articulated shape characters, the ARAP energy is integrated into the linear skinning deformation method [30]. The ARAP energy has also been applied to dynamic shape reconstruction [31, 32]. Yang et al. [33] consider adjusting deformation stiffness using different neighborhood sizes. Their method however is based on voxels, which suffers from high computational costs when the grid is dense, or is unable to represent deformations at fine scales if the grid is coarse. Our method works directly on meshes which also avoids the need of converting between meshes and voxels. We also propose a method to automatically learn adaptive neighborhood sizes. Recent progress has also been made to speed up the ARAP deformation with GPU acceleration [11] and the subspace technique [34] for interactive editing. In this paper, we focus on improving the deformation effectiveness.

3. Algorithm

Similar to traditional ARAP deformation, we assume that an input model is provided with a set of handles. The user then moves the handles to desired locations and the algorithm produces a deformed model which satisfies the handle constraints and keeps geometric details. The fundamental spirit of the ARAP deformation is to deform shapes locally rigidly. The traditional ARAP approach interprets the local area as 1-ring neighborhoods. Adjacent 1-ring neighborhoods share a single common edge. This edge constrains the consistency or smoothness of rigid rotations between adjacent transformations. Instead of using 1-ring neighborhoods, we propose to use general r-ring neighborhoods to define local areas, to allow adjustable stiffness.

3.1. r-ring ARAP energy

Let $N(k, r)$ be the set of vertices and associated edges within the r-ring neighborhood of vertex k, where a vertex $j \in N(k, r)$ if and only if there exists a path connect-



(a) (b)

Figure 2: (a) the 1-ring neighborhood edges of the center vertex, (b) the 2-ring neighborhood edges of the center vertex.

ing vertices k and j with the number of edges no more than r. An edge $(i, j) \in N(k, r)$ if it can be visited by a path containing up to r edges from the vertex k. Figure 2 illustrates 1-ring and 2-ring neighborhoods respectively with edges leading to these vertices highlighted. The vertex and edge set $N(k, r)$ is obtained efficiently using breadth-first search from each vertex k. The r-ring energy is defined as follows:

$$E_r(\mathbf{p}', \mathbf{R}) = \sum_{k=1}^{n} \left\{ \sum_{(i,j)\in N(k,r)} w_{ij} \|(\mathbf{p}_i^{'} - \mathbf{p}_j^{'}) - \mathbf{R}_k(\mathbf{p}_i - \mathbf{p}_j)\|^2 \right\}, \tag{1}$$

where n is the number of vertices, $\mathbf{p}_i$ is the vertex position in the input shape, $\mathbf{p}_i^{'}$ is the vertex position after deformation, and $\mathbf{R}_i$ is the rigid rotation to be optimized in each r-ring neighborhood. $\mathbf{p}' = \{\mathbf{p}_i'\}$ and $\mathbf{R} = \{\mathbf{R}_i\}$ represent the deformed positions and local rotation matrices for all the vertices.

When $r = 1$, this energy formulation is equivalent to the standard ARAP deformation [10]. w_{ij} is the cotangent weight which helps to make the energy insensitive to surface discretization [35] and is defined as:

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

where α_{ij} and β_{ij} are two angles opposite to the edge (i, j).

3.2. Optimization Framework

Given the input model and the handle positions after deformation, we optimize the local rotation matrix $\mathbf{R}_i$ and the deformed position $\mathbf{p}_i^{'}$ for each vertex i iteratively. To make this tractable, two alternating steps are applied in each iteration. In the global step, given the rigid rotations $\mathbf{R}_i$, we optimize the vertex positions $\mathbf{p}_i^{'}$, and in the local step, we optimize the rigid rotations $\mathbf{R}_i$ with the vertex positions $\mathbf{p}_i^{'}$ fixed. To begin

with, the rigid rotation for each r-ring neighborhood is initialized with the identity matrix. We now give details for the global and local steps in the following subsections.

3.2.1. Global Step

Given the optimized rigid rotations $\mathbf{R}_i$, the r-ring ARAP energy becomes a quadratic function w.r.t. the deformed positions. The optimal position for $\mathbf{p}_i^{'}$ can thus be obtained by solving the linear system $\frac{\partial E_r}{\partial \mathbf{p}_i^{'}} = 0$:

$$\frac{\partial E_r}{\partial \mathbf{p}_i^{'}} = \sum_{j \in N(i,1)} \left(\sum_{k:(i,j) \in N(k,r)} 2w_{ij} \left((\mathbf{p}_i^{'} - \mathbf{p}_j^{'}) - \mathbf{R}_k(\mathbf{p}_i - \mathbf{p}_j) \right) \right.$$
$$\left. + \sum_{s:(j,i) \in N(s,r)} -2w_{ji} \left((\mathbf{p}_j^{'} - \mathbf{p}_i^{'}) - \mathbf{R}_s(\mathbf{p}_j - \mathbf{p}_i) \right) \right), \tag{3}$$

where $\{k|(i,j) \in N(k,r)\}$ is the vertex set containing all the vertices whose r-ring neighborhood covers the edge (i,j). Since $w_{ij} = w_{ji}$, $\frac{\partial E_r}{\partial \mathbf{p}_i^{'}}$ can be rewritten as

$$\sum_{j \in N(i,1)} 2w_{ij} \left(2d_{ij}(\mathbf{p}_i^{'} - \mathbf{p}_j^{'}) - \sum_{k:(i,j) \in N(k,r)} \mathbf{R}_k(\mathbf{p}_i - \mathbf{p}_j) \right), \tag{4}$$

where d_{ij} is the number of elements in $\{k|(i,j) \in N(k,r)\}$. The linear system $\frac{\partial E_r}{\partial \mathbf{p}_i^{'}} = 0$ is defined as

$$2d_{ij} \sum_{j \in N(i,1)} w_{ij}(\mathbf{p}_i^{'} - \mathbf{p}_j^{'}) = \sum_{j \in N(i,1)} w_{ij} \sum_{k:(i,j) \in N(k,r)} \mathbf{R}_k(\mathbf{p}_i - \mathbf{p}_j) \tag{5}$$

This induces a linear system $\mathbf{A}\mathbf{p}^{'} = \mathbf{b}$. During deformation, assuming H is the set of handle vertices with user specified positional constraints. For vertex $i \in H$, the specified handle position is $\mathbf{c}_i$. This is equivalent to having a hard constraint $\mathbf{p}_i^{'} = \mathbf{c}_i$. For each $i \in H$, the corresponding i^{th} row and i^{th} column of $\mathbf{A}$ will be set to zero except for the diagonal element where $\mathbf{A}(i,i) = 1$. The i^{th} row of $\mathbf{b}$ is set to $\mathbf{c}_i$.

$\mathbf{A}$ is purely determined by the input model and the set of handle vertices, so it is a fixed matrix during interactive deformation. Since $\mathbf{A}$ is symmetric and semi-definite, we apply Cholesky decomposition to $\mathbf{A}$ in advance and the linear system can be efficiently solved to obtain vertex positions $\mathbf{p}^{'}$ by back substitution during the optimization step. Both the Cholesky decomposition and back substitution are implemented by MATLAB which has been optimized for parallel computing.

3.2.2. Local Step

Given the optimized vertex positions $\mathbf{p}^{'}$, in the local step, the rigid rotation $\mathbf{R}_i$ for each vertex i is optimized as follows. Let $\mathbf{S}_i = \sum_{(j,k) \in N(i,r)} w_{kj}(\mathbf{p}_k - \mathbf{p}_j)(\mathbf{p}_k^{'} - \mathbf{p}_j^{'})^T$. Similar to [10], the optimal rotation can be obtained explicitly. We first apply singular value decomposition (SVD) to $\mathbf{S}_i$, giving $\mathbf{S}_i = \mathbf{U}_i \Sigma_i \mathbf{V}_i^T$. The optimized rigid rotation

$\mathbf{R}_i$ can be obtained as $\mathbf{V}_i{\mathbf{U}_i}^T$. The sign of $\mathbf{U}_i$ corresponding to the smallest singular value should be changed when necessary to make $\det \mathbf{R}_i > 0$. The rigid rotation optimization is independent for each r-ring neighborhood, so this optimization can be straightforwardly accelerated in parallel by OpenMP.

3.2.3. Convergence Condition

In each global/local step, the energy E_r is monotonically decreasing, so the optimization always converges to some local minima. With different r, the value of the optimized energy is also different. To set a consistent termination condition, we normalize the energy difference between the $(t-1)^{\text{th}}$ iteration and the t^{th} iteration with the energy of the $(t-1)^{\text{th}}$ iteration. The optimization is terminated if the following condition is satisfied:

$$\frac{E_r^{(t-1)} - E_r^{(t)}}{E_r^{(t-1)}} < \gamma$$

For all the examples in this paper, the parameter γ is chosen as 10^{-3}.

3.3. Spatially Varying Rigidity

Real-world objects are often composed of different materials with different stiffness. Our approach allows such objects to be simulated within a unified framework by using the r-ring ARAP formulation with different r values specified for different regions. The resulting energy with varying r can be optimized using the same framework as described in Section 3.2.1 and Section 3.2.2. We further developed an intuitive graphical user interface application to help users specify different neighborhood sizes for different regions. It provides a variety of tools. The user can select the current r, and use a paintbrush tool to assign r to surface regions by directly painting on the surface. Alternatively, the user may select a region and use a flood fill (paint bucket) tool to assign r to the whole selected region. A simple example is shown in Figure 8. The bar is made with two different materials with the softer part rendered in orange and the harder part in gray. As shown in the results, enlarging r increases rigidity.

3.4. Learning Spatially Varying r from Deforming Shapes

Instead of specifying r by manual painting, when a set of deforming shapes is available, we propose an automatic method to assign suitable spatially varying r across the surface. Intuitively, larger r leads to more rigid deformation, and thus tends to preserve details better. However, when r is set to be too large, the shape can be locally too rigid, and thus results in a large deformation error, which can be efficiently estimated using the As-Rigid-As-Possible energy.

Assume for each vertex k, we set the neighborhood size r_k to an integer in the range of $[r_{\min}, r_{\max}]$. We further assume that M models are available, and the position of the i^{th} vertex on the m^{th} model is denoted as $\mathbf{p}_i^m$. We take the first model as the reference model, and for an arbitrary model m ($m = 2, 3, \ldots, M$), we can work out the r-ring ARAP energy for vertex k as follows:

$$E(k,r,m) = \sum_{(i,j)\in N(k,r)} w_{ij} \|(\mathbf{p}_i^m - \mathbf{p}_j^m) - \mathbf{R}_k(\mathbf{p}_i^1 - \mathbf{p}_j^1)\|^2$$

We normalize the energy $E(k,r,m)$ to make it scale invariant :

$$\tilde{E}(k,r,m) = \frac{1}{A(k,r)} E(k,r,m)$$

where $A(k,r)$ is the sum of the Voronoi areas of all the vertices in $N(k,r)$ on the reference model.

To measure the overall deformation for the k^{th} vertex, we take the mean $\tilde{E}(k,r,m)$ as ARAP energy for vertex k with r-ring neighborhood:

$$E_{ARAP}(k,r) = \frac{1}{M-1} \sum_{2\leq m\leq M} \tilde{E}(k,r,m)$$

To penalize locally non-rigid deformation, we favor larger r, and hence introduce the non-rigid energy as:

$$E_{non-rigid}(k,r) = r_{\max} - r_k$$

For each vertex k, the neighborhood size r_k is obtained by minimizing the energy combining both terms:

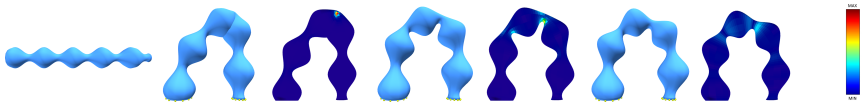
$$r_k = \arg\min_r \left(E_{ARAP}(k,r) + \omega E_{non-rigid}(k,r)\right)$$

ω is a globally adjustable parameter to control the preference of rigidity to deformation error. Thanks to the normalization, we find a default set of the parameters works well for a wide range of shapes. We set $r_{\min} = 1$, $r_{\max} = 6$ and $\omega = 0.3$ in all our experiments.

4. Results and Discussions

In this section, we show various deformation results using our approach including a single neighborhood size and mixed neighborhood sizes, and compare our results with state-of-the-art deformation methods. The experiments were carried out on a computer with an Intel i7-2600 CPU and 8GB RAM. We use yellow dots to indicate the deformation handles.

Timing & Convergence. As discussed before, the energy of our deformation approach is monotonically decreasing so it always converges to some local minima. The running time of our method includes the off-line step and the on-line step. The off-line step involves the breadth first search (BFS) to obtain the r-ring neighborhood $N(i,r)$ for each vertex i, and the predecomposition of the sparse matrix $\mathbf{A}$. These can be performed independent of the handle positions and thus only need to be performed once during the interactive deformation process. The online step mainly consists of the time



(a) (b) (c) (d) (e) (f) (g)

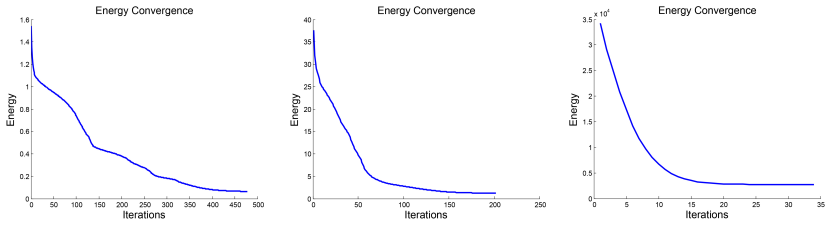
Figure 3: Deformation results of different neighborhood sizes (r). (a) the input shape, (b)(d)(f) the results with 1-ring (equivalent to ARAP deformation [10]), 2-ring and 10-ring neighborhoods, (c)(e)(g) the color coded energy distribution of (b)(d)(f).

Ring Number	BFS (ms)	Cholesky (ms)	Global (ms)	Local (ms)	#Iterations	Total Time (s)
1-ring	3.65	26.87	22.96	2.20	477	12.03
2-ring	8.91	26.65	24.57	5.45	201	6.06
4-ring	28.68	19.75	39.11	20.43	124	7.43
6-ring	63.47	19.02	69.21	45.95	63	7.34
10-ring	136.09	19.50	129.17	103.48	33	7.70

Table 1: Statistics of the deformation running times for the example in Figure 3.

for global and local optimization. The detailed running times for the deformation example in Figure 3 are shown in Table 1. Note that when 1-ring neighborhood is used, our method reverts to the standard ARAP deformation [10]. The model involved contains 1154 vertices. For the same model with different neighborhood sizes r, the size of the linear system is the same, which is equal to the number of vertices. As a result, the running times of Cholesky decomposition, back substitution and SVD decomposition are similar. The major differences are the times for using the BFS to build r-ring neighborhoods, calculating the matrix $\mathbf{S}_i$ in the local step and the vector $\mathbf{b}$ in the global step. With the increasing neighborhood size r, the running times for BFS, local optimization and global optimization are increasing while the time for Cholesky decomposition is decreasing as shown in Table 1.

With increasing r, the material becomes more rigid. The consistency of adjacent rigid rotations is enhanced so the flexility of rigid rotations is reduced. Thus fewer iterations are needed to converge. Figure 4 shows how the r-ring ARAP energy converges over iterations for different neighborhood sizes. Figure 5 further shows the number of



(a) (b) (c)

Figure 4: Energy convergence curves of the example in Figure 3 with (a) 1-ring, (b) 2-ring, (c) 10-ring.

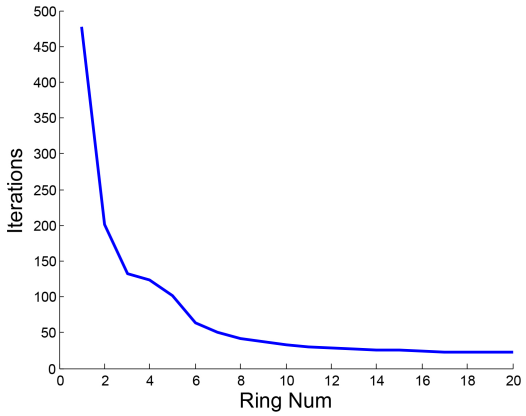


Figure 5: The number of iterations (y-axis) needed for convergence w.r.t. the neighbood size r (x-axis), according to the same convergence condition.

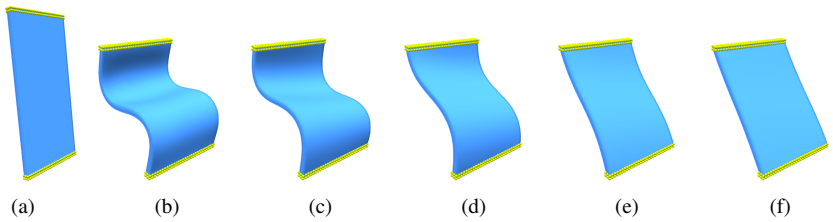
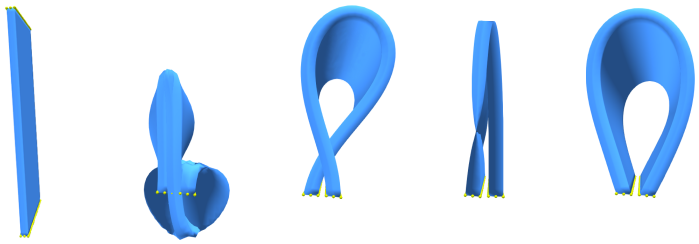


Figure 6: Deformation results of different neighborhood sizes. (a) the input shape, (b) 1-ring result, (c) 5-ring result, (d) 10-ring result, (e) 15-ring result, (f) 20-ring result.

iterations required for the energy to converge, based on the same convergence condition (Section 3.2.3). As it takes longer for each iteration and the number of iterations reduces with increasing r, the running times are fairly consistent from 2-ring to 10-ring. For the example in Figure 3 the running time is between 6s-7s, which is about half of the time as traditional ARAP deformation (equivalent to setting $r = 1$). For the human shape with $12.5K$ vertices shown in Figure 9, the online optimization takes about 10.2s with mixed r-ring neighborhood sizes.

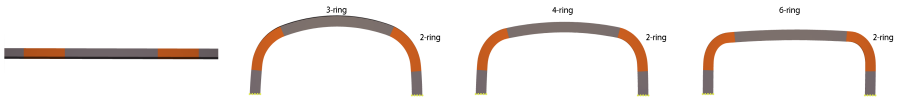
The running time for automatically learning neighborhood sizes from deforming model dataset is mainly spent on calculating general ARAP energies with different neighborhood sizes around each vertex, so is proportional to the vertex number of each model, as well as the total number of models. The running time is about 0.68 ms per vertex per model, to calculate energies from 1-ring to 6-ring neighborhoods. For the human shape in Figure 9 with $12.5K$ vertices and 71 models, the one-off neighborhood size selection algorithm takes about 10 minutes.

Results with global change of r. Figure 3 demonstrates deformation results with one end of the shape bent by $180°$. The traditional ARAP deformation method (equivalent to using 1-ring neighborhood) [10] produces self-intersection artifacts. With such



(a) (b) (c) (d) (e)

Figure 7: Comparison of deformation results. (a) the input shape, (b) ARAP deformation result [10], (c) SR-ARAP deformation result [12], (d) deformation result of [36], (e) result of our approach with 6-ring neighborhood.



(a) (b) (c) (d)

Figure 8: Deformation results of different neighborhood sizes. (a) the input shape, (b) mixed 3-ring and 2-ring, (c) mixed 4-ring and 2-ring, (d) mixed 6-ring and 2-ring.

large deformation, the distortions cannot be distributed sufficiently uniformly. With larger neighborhood sizes, the rigid consistency is strengthened. The distortions are propagated much more uniformly. We show the deformation results with 2-ring and 10-ring where no self-intersections are generated. To visualize how the energy varies locally over the surfaces and to account for the difference in absolute energy values, we show color coding of energy difference between adjacent r-ring neighborhoods. It can be clearly seen that in the traditional 1-ring case, significant energy change is concentrated on small regions, and the energy is distributed more uniformly with the increasing neighborhood size.

Figure 6 shows the deformation results with the same user constraints but changing neighborhood size r. The bar tends to become more rigid with increasing r. The neighborhood sizes (1-ring, 5-ring, 10-ring and 15-ring) are chosen to demonstrate typical controllable rigidity. The 1-ring deformation result looks like elastic plastic material whereas the 15-ring deformation result looks more like metal. With a single adjustable parameter, the user can change the material properties freely and intuitively.

We show further example with substantial deformation and compare our results with state-of-the-art methods. The example is shown in Figure 7, where our method produces a natural deformation result while state-of-the-art methods produce results with artifacts, including self-intersections and over blended distortions. These examples demonstrate that by using a larger neighborhood size, our method can avoid deformation artifacts typically appearing in existing methods which are induced by the local minimum nature of optimization.

Results with spatially varying neighborhood size r. Our algorithm also allows

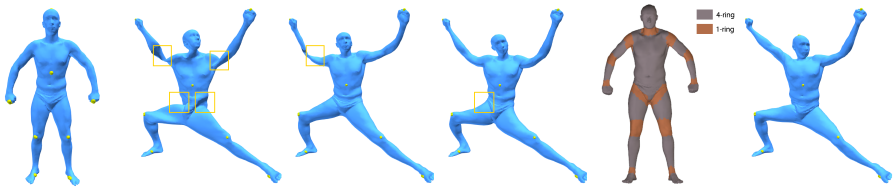


Figure 9: Comparison of deformation results. (a) the input shape, (b) ARAP deformation result [10], (c) SR-ARAP deformation result [12], (d) deformation result of [36], (e) r-ring neighborhoods specified by the user, (f) result of our approach with mixed r-ring neighboorhoods.

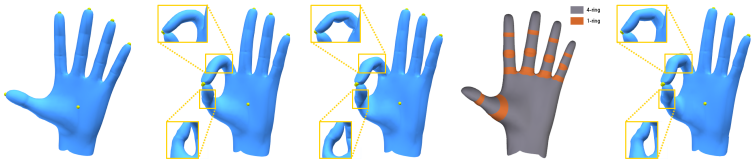


Figure 10: Deformation result comparisons. (a) the input shape, (b) deformation result of [36], (c) result of ARAP deformation [10], (d) neighborhood sizes r painted by the user, (e) deformation results with mixed r-ring neighborhoods.

the user to specify the material properties by a paintbrush. As shown in Figure 8, the thin bar is painted with two different neighborhood sizes. The smaller neighborhood size r is specified for softer areas where more bending is allowed. With the increasing r, the middle part of the bar tends to be more rigid.

In Figures 9, 10 and 11, the user specifies the rigidity of shape regions according to the intrinsic properties. The joint area tends to be much more flexible for articulated shape deformation. As shown in the results, the shape deformation results with mixed r are much more natural than the previous state-of-the-art methods [10, 12, 36]. As highlighted in the yellow rectangles, deformation artifacts including self intersections, excessive twisting and unnatural distortions occur in the deformation results of previous methods. In Figure 9, muscle contraction appears in the deformation results of

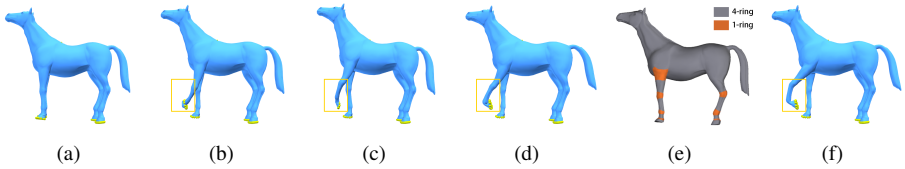


Figure 11: Deformation result comparisons. (a) The input shape, (b) SR-ARAP deformation results [12], (c) deformation result of [36], (d) ARAP deformation results [10],(e) neighborhood sizes r painted by the user, (f) deformation results with mixed r-ring neighborhoods.

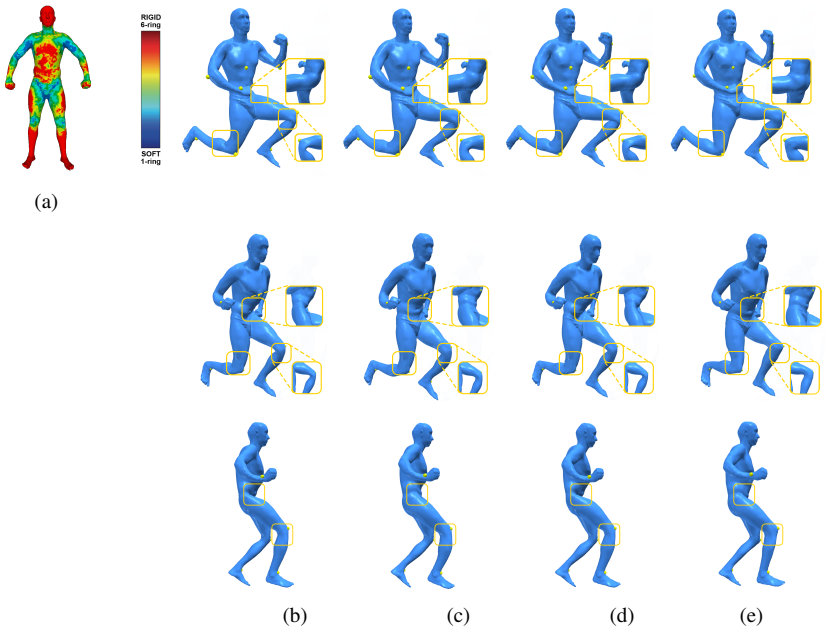


Figure 12: Comparison of deformation with our automatic neighborhood size selection and alternative methods. (a) color coding illustrating the neighborhood size (1–6 corresponding to blue to red), (b) SR-ARAP deformation results [12], (c) deformation results of [36], (d) ARAP deformation results [10], (e) our deformation results using the learned adaptive neighborhood size.

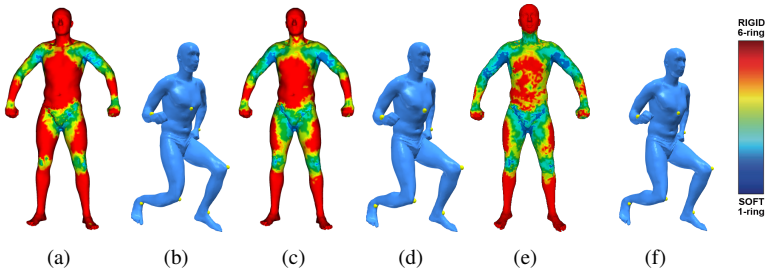


Figure 13: Deformation result comparisons with neighborhood sizes automatically learned using (a-b) 7, (c-d) 30 and (e-f) 71 examples from the SCAPE dataset. (a)(c)(e) color coding illustrating the neighborhood size, (b)(d)(f) corresponding deformation results.

[10, 12, 36] which looks unrealistic. The bending areas of these method in the arm fail to be located at elbow joints. The deformed human shape of our method is free of these artifacts. In Figure 10, the bent index finger of [10] looks like elastic plastic without joints. The thumb of [36] is squashed. With guiding rigidity distribution, the deformation result of our method looks much more natural. Similar unnatural distortions of [10, 12, 36] also appear in Figure 11. Compared with these methods, our method makes natural and reasonable deformations.

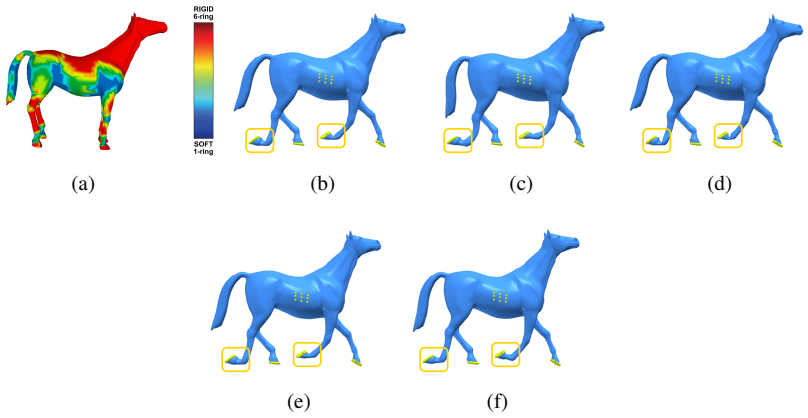


Figure 14: Comparison of deformation using the horse model with our automatic neighborhood size selection and alternative methods. (a) color coding illustrating the neighborhood size (1–6 corresponding to blue to red), (b) SR-ARAP deformation results [12], (c) deformation result of [36], (d) ARAP deformation results [10], (e) our extended 3-ring ARAP deformation results, (f) our deformation results using the learned adaptive neighborhood size.

Results with automatically selected neighborhood size r. We now show the results obtained using automatic neighborhood size selection, and compare them with alternative methods. For this purpose, we need a collection of shapes with the same connectivity, which many existing datasets satisfy (or can be achieved by consistent remeshing).

In Figure 12, we show results based on the SCAPE (Shape Completion and Animation of People) dataset [37] of deforming human body. The automatically selected neighborhood sizes using the whole dataset of 71 models are illustrated using color-coding in Figure 12(a). It effectively identifies the rigid parts (such as the head) and regions which are locally non-rigid (such as joints) and assigns suitable neighborhood sizes. We show three deformation examples with user-specified handles highlighted. For all examples, results of our method (Figure 12(e)) with learned adaptive neighborhood sizes preserve details for the rigid parts well while allowing non-rigid parts to deform flexibly to produce natural deformation results. On the contrary, the three state-of-the-art methods [10, 12, 36] do not preserve the rigid parts (e.g. legs) and produce distortions around joints.

To test the influence of the model database on the results, we further use the first 7 and 30 models from the SCAPE dataset. The results are shown in Figure 13. When the number of examples is small (with only 7 models), it is not sufficient to capture all the possible non-rigid deformations. As a result, the deformed right leg bones show clear bending as the joint is not properly recognized. When 30 models are used for learning, the neighborhood size distribution is very similar to using the full dataset, and the result looks plausible.

Another example is shown in Figure 14, using the horse dataset from [38]. Existing state-of-the-art methods [10, 12, 36] have artifacts such as bent legs (c) and smoothed out joints (b-d). Our method with a fixed neighborhood size of 3 (e) while better

preserves details than the traditional ARAP, still fails to preserve the shapes of hooves well enough (as they should be more rigid) and somewhat smoothes out the joints (as they should be more flexible). Our method with learned adaptive neighbourhood produces results look natural without such artifacts.

5. Conclusions and Future Work

In this paper, we extend the ARAP deformation model from 1-ring neighborhoods to general r-ring, which allows a series of natural deformation to be achieved, mimicking objects made up with different materials. We further consider using spatially varying neighborhood sizes that adapt to the local rigidity, either specified manually using an intuitive paintbrush interface, or learned automatically from a set of deforming shapes. Such adaptive neighborhood sizes help to further improve flexibility and allow more natural deformations to be achieved.

Our current implementation is purely CPU-based. Although with OpenMP-based multithreading, it is sufficient for interactive deformation, it still cannot run in real time. The algorithm can potentially be further optimized by GPGPU computing. The local optimization of estimating the local rigid rotations $\mathbf{R_i}$ and the global optimization of solving the predecomposed equations can be parallelized by GPGPU.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61502453 and No. 61611130215), Royal Society-Newton Mobility Grant (No. IE150731), CCF-Tencent Open Research Fund (No. AGR20160118), Knowledge Innovation Program of the Institute of Computing Technology of the Chinese Academy of Sciences (ICT20166040).

References

[1] J. Lewis, M. Cordner, N. Fong, Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation, in: Proceedings of ACM SIGGRAH, 2000, pp. 165–172.

[2] H.-B. Yan, S.-M. Hu, R. R. Martin, Y.-L. Yang, Shape deformation using a skeleton to drive simplex transformations, IEEE Transaction on Visualization and Computer Graphics 14 (3) (2008) 693–706.

[3] T. Ju, S. Schaefer, J. Warren, Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics 24 (3) (2005) 561–566.

[4] P. Joshi, M. Meyer, T. DeRose, B. Green, T. Sanocki, Harmonic coordinates for character articulation, ACM Transactions on Graphics 26 (3) (2007) Article No. 71.

[5] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, H.-P. Seidel, Laplacian surface editing, in: Proceedings of Eurographics Symposium on Geometry Processing, 2004, pp. 175–184.

[6] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum, Mesh editing with Poisson-based gradient field manipulation, in: Proceedings of ACM SIGGRAPH, 2004, pp. 644–651.

[7] O. K.-C. Au, C.-L. Tai, L. Liu, H. Fu, Dual laplacian editing for meshes, IEEE Transactions on Visualization and Computer Graphics 12 (3) (2006) 386–395.

[8] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, H.-Y. Shum, Large mesh deformation using the volumetric graph laplacian, ACM Transactions on Graphics 24 (3) (2005) 496–503.

[9] Y. Lipman, O. Sorkine, D. Levin, D. Cohen-Or, Linear rotation-invariant coordinates for meshes, in: Proceedings of ACM SIGGRAPH, 2005, pp. 479–487.

[10] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: Proceedings of Eurographics Symposium on Geometry Processing, 2007, pp. 109–116.

[11] M. Zollhöfer, E. Sert, G. Greiner, J. Süßmuth, GPU based ARAP deformation using volumetric lattices, in: Eurographics (Short Papers), 2012, pp. 85–88.

[12] Z. Levi, C. Gotsman, Smooth rotation enhanced as-rigid-as-possible mesh animation, IEEE Transactions on Visualization and Computer Graphics 21 (2) (2015) 264–277.

[13] M. Botsch, O. Sorkine, On linear variational surface deformation methods, IEEE Transactions on Visualization and Computer Graphics 14 (1) (2008) 213–230.

[14] D. Cohen-Or, Space deformations, surface deformations and the opportunities in-between, Journal of Computer Science and Technology 24 (1) (2009) 2–5.

[15] J. Gain, D. Bechmann, A survey of spatial deformation from a user-centered perspective, ACM Transactions on Graphics 27 (4) (2008) Article No. 107.

[16] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically deformable models, in: Proceedings of ACM SIGGRAPH, 1987, pp. 205–214.

[17] D. L. James, D. K. Pai, ArtDefo: accurate real time deformable objects, in: Proceedings of ACM SIGGRAPH, 1999, pp. 65–72.

[18] Y. Lipman, D. Levin, D. Cohen-Or, Green coordinates, ACM Transactions on Graphics 27 (3) (2008) Article No. 78.

[19] T. Popa, D. Julius, A. Sheffer, Interactive and linear material aware deformations, International Journal of Shape modeling.

[20] A. Murai, Q. Y. Hong, K. Yamane, J. K. Hodgins, Dynamic skin deformation simulation using musculoskeletal model and soft tissue dynamics, Computational Visual Media (2016) 1–12.

[21] M. Alexa, D. Cohen-Or, D. Levin, As-rigid-as-possible shape interpolation, in: Proceedings of ACM SIGGAPH, 2000, pp. 157–164.

[22] L. Ya-Shu, H.-B. Yan, R. R. Martin, As-rigid-as possible surface morphing, Journal of computer science and technology 31 (4) (2012) 77:1–77:10.

[23] T. Igarashi, T. Moscovich, J. F. Hughes, As-rigid-as-possible shape manipulation, in: Proceedings of ACM SIGGRAPH, 2005, pp. 1134–1141.

[24] L. Gao, G.-X. Zhang, Y.-K. Lai, Lp shape deformation, Science China Information Sciences 55 (5) (2012) 983–993.

[25] L. Gao, Y. Lai, Q. Huang, S. Hu, A data-driven approach to realistic shape morphing, Comput. Graph. Forum 32 (2) (2013) 449–457.

[26] Q.-X. Huang, B. Adams, M. Wicke, L. J. Guibas, Non-rigid registration under isometric deformations, in: Proceedings of the Symposium on Geometry Processing, SGP '08, 2008, pp. 1449–1457.

[27] L. Liu, L. Zhang, Y. Xu, C. Gotsman, S. J. Gortler, A local/global approach to mesh parameterization, in: Proceedings of the Symposium on Geometry Processing, 2008, pp. 1495–1504.

[28] I. Chao, U. Pinkall, P. Sanan, P. Schröder, A simple geometric model for elastic deformations, ACM Trans. Graph. 29 (4) (2010) 38:1–38:6.

[29] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, S. Xia, Efficient and flexible deformation representation for data-driven surface modeling, ACM Trans. Graph. 35 (5) (2016) 158:1–158:17.

[30] A. Jacobson, I. Baran, L. Kavan, J. Popović, O. Sorkine, Fast automatic skinning transformations, ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 31 (4) (2012) 77:1–77:10.

[31] M. Zollhöfer, M. Nießner, S. Izadi, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, M. Stamminger, Real-time non-rigid reconstruction using an RGB-D camera, ACM Transactions on Graphics (TOG) 33 (4).

[32] R. A. Newcombe, D. Fox, S. M. Seitz, Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time, in: CVPR, 2015.

[33] W. Yang, J. Feng, X. Jin, Shape deformation with tunable stiffness, The Visual Computer 24 (7-9) (2008) 495–503.

[34] Y. Wang, A. Jacobson, J. Barbič, L. Kavan, Linear subspace design for real-time shape deformation, ACM Transactions on Graphics (TOG) 34 (4).

[35] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete diffential-geometry operators for triangulated 2-manifolds, in: Visualization and Mathematics III, 2003, pp. 35–57.

[36] R. W. Sumner, J. Schmid, M. Pauly, Embedded deformation for shape manipulation, ACM Trans. Graph. 26 (3).

[37] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, SCAPE: Shape completion and animation of people, ACM Trans. Graph. 24 (3) (2005) 408–416.

[38] P. J. Sumner, R. W., Deformation transfer for triangle meshes, ACM Trans. Graph. 23 (3) (2004) 399–405.