Visual Comput (2008) 24: 495–503
DOI 10.1007/s00371-008-0230-3

ORIGINAL ARTICLE

Wenwu Yang
Jieqing Feng
Xiaogang Jin

Shape deformation with tunable stiffness

Published online: 29 May 2008

W. Yang · J. Feng (✉) · X. Jin
State Key Lab of CAD&CG,
Zhejiang University,
Hangzhou 310027, P.R. China
{wwyang, jqfeng, jin}@cad.zju.edu.cn

Abstract The paper presents a 2D or 3D shape deformation method which incorporates global and local stiffness controls. First, a geometric object is embedded into a regular lattice and then the deformation is conducted on the lattice; thus, the method is independent of the underlying object representation. The lattice cells are organized as overlapping local rigid regions, and the region width could be regarded as a means of the global lattice stiffness control. For each region, there is a local stiffness coefficient to control the lattice deformation locally. During the deformation a nonlinear objective function is optimized to achieve the natural lattice deformation with the prescribed global and local stiffnesses. Then, the lattice deformation is passed to the embedded object through bilinear or trilinear interpolation. In this way we can deform the object in a more physically plausible way with tunable stiffness. Experimental results show that the method is intuitive and flexible.

Keywords Shape deformation · Stiffness · Rigidity

1 Introduction

Shape deformation is a useful tool of shape modeling and animation in computer graphics. Direct shape manipulation is particularly attractive since it provides an intuitive and flexible way to edit the object by specifying a few position constraints on the geometry of the model. Other kinds of constraints such as area or volume preservation of 2D or 3D shapes [10, 30] and the free intersection [32] etc. have also been taken into account to achieve more natural deformations. Meanwhile, in the real world, objects usually are made of different types of materials and the materials may also be inhomogeneous. So, the object deformations will manifest different stiffness behaviors. Thus, the deformation taking into account the stiffness will be more flexible and physically plausible.

This paper presents a 2D or 3D shape deformation method that incorporates the global and local stiffness properties of the object. The incorporation of stiffness control provides an intuitive way for a user to specify the rigidity of the object and allows us to mimic shape deformations composed of different materials. The method deforms the shape under the position constraints with the prescribed global and local stiffness parameters while minimizing distortion. First, the shape is embedded into a regular lattice and the lattice cells are organized as overlapping local rigid regions with a specified width. Increasing (decreasing) the region width is equivalent to increasing (decreasing) the global stiffness of the shape,

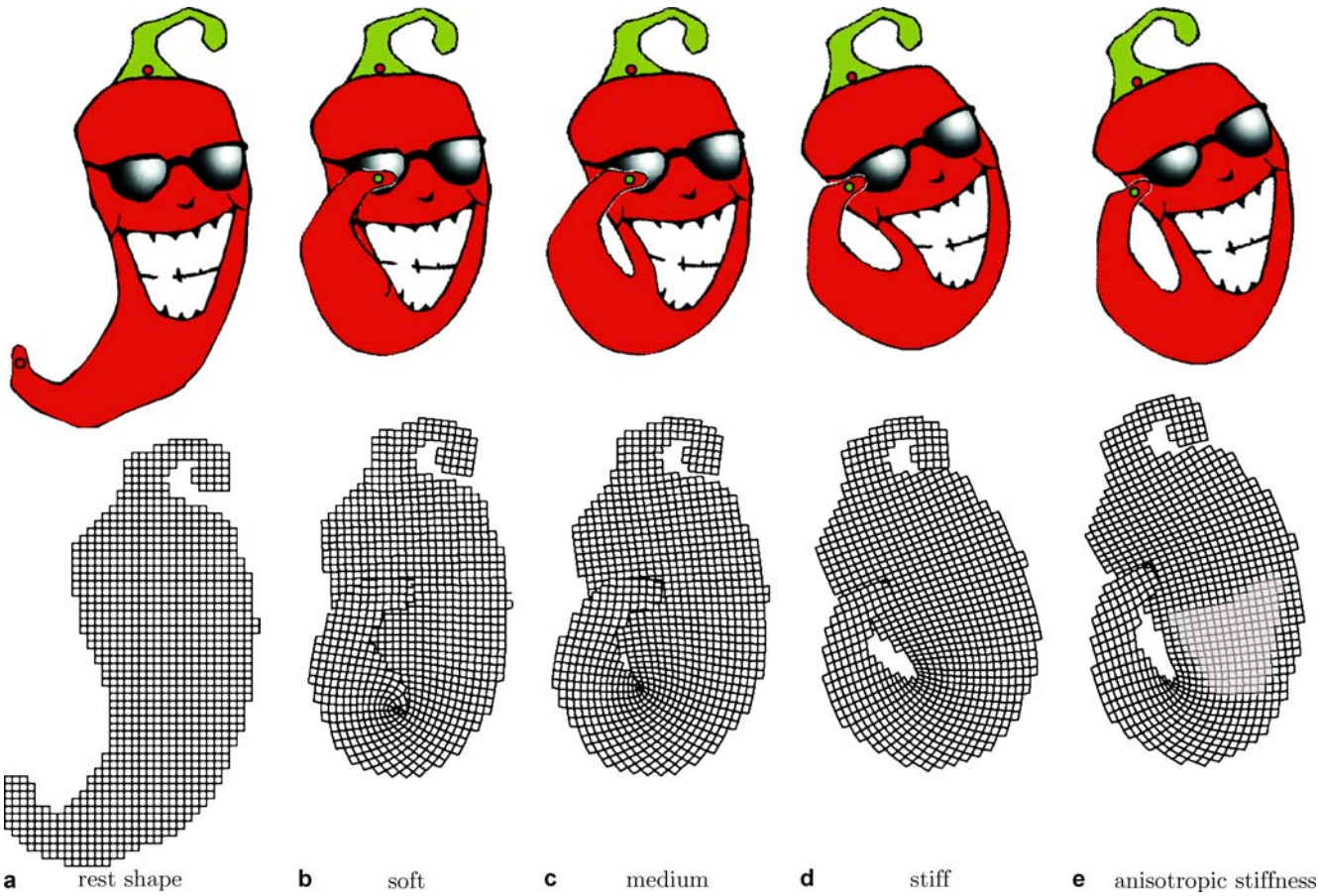

Fig. 1a–e. A capsicum-like cartoon shape (**a**) is deformed under the same constraint with different global stiffnesses, where **b–d** correspond to the lattice region half-widths $w = 1$, 3 and 6, respectively. The *top row* are deformation results and the *bottom row* are the corresponding lattices. The mouth distortion in (**d**) is alleviated in (**e**) by increasing the local stiffness at the mouth region, which is indicated by the *gray part* in the lattice of (**e**)

which effectively mimics the stiff (soft) object deformations (see Fig. 1b–d). A penalty coefficient for the distortion of each local rigid region defines the local stiffness and can also be adjusted to reflect the anisotropic stiffness (see Fig. 1e). To achieve a physically plausible deformation, a nonlinear energy function that measures the deviation of each rigid region transformation from a true rigid transformation (i.e. translation, rotation or their combination) is designed and will be minimized for the desirable deformation results. Meanwhile, the lattice regularity could be exploited to improve the optimization efficiency by using the fast summation technique [21].

2 Related work

As an important shape editing and animation technique, shape deformation has been well studied for many years. The relevant works are reviewed here.

Free-form deformation (abbreviated as FFD) [24] is the most prevalent shape deformation method. The FFD is conducted on a parametric volume in which the object is embedded; thus, it separates shape editing complexity from geometry complexity and shape representation. However, the FFD is a shape editing tool geometrically without any considerations of the object material ingredients. Thus, it could not be applied to the stiffness-awareness deformation directly [20].

For articulated models with jointed structures, it would be better to use the skeleton-driven deformation techniques [15]. The skeleton-based approach can provide a binary gradation of stiffness. However, automatically determining the skeleton configuration is not a trivial work and most of the skeleton-based methods are not suitable for non-articulated objects.

Physical-based simulation can naturally [7, 8, 13, 18, 19, 23, 29] incorporate the material properties into the shape deformation and tends to produce realistic results with physical accuracy and correctness. However, the

physical equations which govern the deformation are usually complicated and expensive to be solved. Furthermore, it is not a trivial work to find an intuitive control means to tune the physical equations' parameters for desirable deformation effects.

Geometry-based differential domain approaches [27, 31] can achieve detail-preserving deformation results through optimizing the local transformations and preserving the differential coordinates simultaneously. It is a nonlinear optimization problem since either 2D or 3D rotation transformations cannot be expressed as a linear function of planar or space position. For the sake of the computational efficiency, the rotation transformation is linearly approximated by local linearization [11, 27], transformation propagation [31, 32] or transformation interpolation [16]. However, the linear approximation may lead to a suboptimal result corresponding to undesirable shape distortions, etc. [5]. The nonlinear approaches can effectively avoid the suboptimal results and achieve high-quality deformations by an iterative optimization [2, 3, 14, 30]. Usually, these nonlinear methods solve the deformations on specified subspaces for the sake of efficiency and stability [1, 4, 10, 28].

Some geometry-based methods [4, 11, 20, 26] can adjust the local stiffness of the object in the deformation. The method in [26] defines an overlapping local cell at each mesh vertex as one-ring neighboring vertices, and keeps the transformation of each cell as rigid as possible during the deformation. By adjusting the size of local cells, the overall rigidity of the object can also be adjusted. However, the computational costs of the deformation algorithm will increase with the increase of the local cell size. Therefore, the method is more appropriate to simulate soft objects rather than stiff ones. In addition, the method may generate results with inconsistent constraints under large deformations, especially for the 'stiff' model (see Fig. 4). The inconsistency could be alleviated by imposing additional constraints, which needs more user interactions.

The proposed deformation method incorporates the global and local stiffness controls but is free of the above problems in [26]. Unlike the method in [26] primarily designed for mesh surfaces, the method presented here is independent of shape representation since it deforms the lattice instead of the geometry itself.

3 Shape deformation with tunable stiffness

In this section, we first introduce the lattice definition and some notation. Then, we describe how to design the nonlinear deformation objective function on the overlapping regions with the specified stiffness. An efficient and robust iterative scheme is adopted to solve the above nonlinear optimization problem. Finally, an intuitive means is designed to tune the local and global object stiffnesses for the physically plausible results.

3.1 Shape discretization and local rigid regions

The object could be 2D or 3D. It is first voxelized as a regular lattice of 2D square or 3D cubic cells by employing the voxelization method [12], as illustrated in Fig. 3. The object is then embedded into the generated lattice by the bilinear or trilinear mapping. Let $\mathcal{L}$ denote the lattice with n nodes, and $\boldsymbol{p}_i$ ($\boldsymbol{q}_i$) be the initial (deformed) node position. For each node i, its w-ring neighborhood set $\mathcal{R}_i$ comprises the nodes whose chess board distance to the node i is not greater than w, which is shown in Fig. 2.

Having the object voxelized, a local rigid region is defined at each lattice node as its w-ring neighborhood, and w is also called the region half-width in the paper. Here, the local rigid region definition is similar to that in [21] where the lattice regions are used for real-time large-scale dynamic simulation of the embedded object. Intuitively, a stiff model is more difficult to be bent or stretched than the soft model.

Obviously, the set of indices of all local rigid regions to which the node i belongs is equivalent to $\mathcal{R}_i$. In gen-

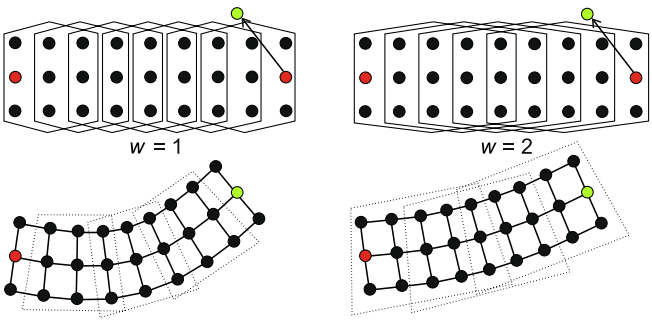

Fig. 2. The region width is adopted as the global stiffness control means. Each *hexagon* in the *top row* represents a local rigid region. The *bottom row* shows the corresponding deformation results under the same constraint where the *right-hand* one ($w = 2$) is more rigid than the *left-hand* one ($w = 1$)

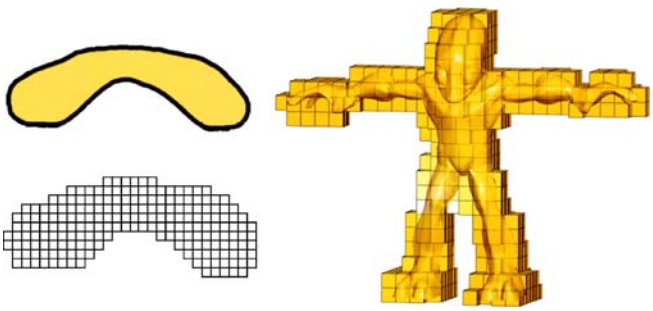

Fig. 3. The 2D or 3D object is voxelized as a regular lattice. The lattice cell is a square in 2D or a cube in 3D

eral, a local rigid region is a square in 2D (cube in 3D) with side length $2w+1$ except for the boundary cases. Obviously, the adjacent local rigid regions are overlapped together. They seem to be glued together for the sake of the smooth lattice deformation. And, with the increase of the half-width w, the neighboring local rigid regions will be more tightly tied, such that the local rigid regions are more difficult to be stretched or bent (see Fig. 2). Thus, the half-width w could be regarded as the global stiffness parameter, i.e. a large value for a stiff object while a small value for a soft object.

3.2 Deformation energy

The algorithm defines a nonlinear energy function to preserve the rigidity of local rigid regions during the shape deformation. We first introduce the rigid transformation of a local rigid region.

Given a local rigid region $\mathcal{R}_i$ at the lattice node i, the initial and deformed positions of the node $j \in \mathcal{R}_i$ are $\boldsymbol{p}_j$ and $\boldsymbol{q}_j$, respectively. If the deformation of the local rigid region is rigid, the optimal rigid transformation $\boldsymbol{A}_i$ is found by matching all initial node positions $\{\boldsymbol{p}_j\}_{j\in\mathcal{R}_i}$ to the deformed positions $\{\boldsymbol{q}_j\}_{j\in\mathcal{R}_i}$ such that

$$\boldsymbol{q}_j - \boldsymbol{q}_c^i = \boldsymbol{A}_i\big(\boldsymbol{p}_j - \boldsymbol{p}_c^i\big), \quad \forall j \in \mathcal{R}_i, \tag{1}$$

where $\boldsymbol{p}_c^i$ and $\boldsymbol{q}_c^i$ are the initial and deformed rotation centers. Since the deformation could not be rigorously rigid, an optimal rigid transformation $\boldsymbol{A}_i$ is found to fit Eq. 1 in a least-square sense, i.e. minimizing

$$E(\mathcal{R}_i) = \sum_{j\in\mathcal{R}_i} \big\|\boldsymbol{q}_j - \boldsymbol{q}_c^i - \boldsymbol{A}_i\big(\boldsymbol{p}_j - \boldsymbol{p}_c^i\big)\big\|^2. \tag{2}$$

It is a shape matching problem [9]. In the 2D case, $\boldsymbol{A}_i$ has an analytical expression[22], i.e.

$$\boldsymbol{A}_i = \frac{1}{\mu_s} \sum_{j\in\mathcal{R}_i} \begin{pmatrix} \hat{\boldsymbol{p}}_j & -\hat{\boldsymbol{p}}_j^{\perp} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{q}}_j^{\mathrm{T}} \\ -\hat{\boldsymbol{q}}_j^{\perp\mathrm{T}} \end{pmatrix}, \tag{3}$$

where

$$\mu_s = \sqrt{\Big(\sum_j \hat{\boldsymbol{q}}_j^{\mathrm{T}} \hat{\boldsymbol{p}}_j\Big)^2 + \Big(\sum_j \hat{\boldsymbol{q}}_j^{\mathrm{T}} \hat{\boldsymbol{p}}_j^{\perp}\Big)^2} \tag{4}$$

with $\hat{\boldsymbol{p}}_j = \boldsymbol{p}_j - \boldsymbol{p}_c^i$ and $\hat{\boldsymbol{q}}_j = \boldsymbol{q}_j - \boldsymbol{q}_c^i$, where $\perp$ is a 2D vector operator such that $(x, y)^{\perp} = (-y, x)$. Although there is no analytical solution for the above $\boldsymbol{A}_i$ in the 3D case [17], a least-square solution of $\boldsymbol{A}_i$ can be expressed as the rotational part of

$$\boldsymbol{A}_t = \sum_j \big(\boldsymbol{q}_j - \boldsymbol{q}_c^i\big)\big(\boldsymbol{p}_j - \boldsymbol{p}_c^i\big)^{\mathrm{T}}. \tag{5}$$

The rotational part of $\boldsymbol{A}_t$ can be obtained by using the polar decomposition $\boldsymbol{A}_t = \boldsymbol{RS}$, where $\boldsymbol{R}$ is an orthogonal identity matrix and $\boldsymbol{S}$ is a diagonally symmetric matrix [25]. Since the diagonally symmetric matrix $\boldsymbol{S}$ includes only stretching, the matrix $\boldsymbol{R}$ should be the rotational part.

To approximate the rigid transformations of all overlapping local rigid regions, a global energy function is defined as a weighted summation of the deviations of all region transformations from their optimal rigid transformations:

$$\begin{aligned} E(\mathcal{L}) &= \sum_i^n w_i E(\mathcal{R}_i) \\ &= \sum_i^n w_i \sum_{j\in\mathcal{R}_i} \big\|\boldsymbol{q}_j - \boldsymbol{q}_c^i - \boldsymbol{A}_i\big(\boldsymbol{p}_j - \boldsymbol{p}_c^i\big)\big\|^2. \end{aligned} \tag{6}$$

The weight w_i is a penalty factor for the deviation of each local rigid region transformation, and could be regarded as a local stiffness control parameter, as illustrated in Figs. 1e and 9.

The energy formulation of Eq. 6 is similar to that in [26] for the rigidity of the local cell at each mesh vertex. The subtle, but important, difference is that the initial and deformed rotation centers ($\boldsymbol{p}_c^i$ and $\boldsymbol{q}_c^i$) are directly selected as the cell's centers (which is equivalent to $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ in our formulation) in [26], whereas our algorithm specifies them in a more quasi-physical way. As described above, each local rigid region transformation is expected to be as rigid as possible; thus, a local rigid region could be assumed as a rigid body unit where all nodes are transformed as a whole. With this assumption it is obvious that if a constraint node in the local rigid region is translated or fixed, the other nodes in this region should follow the same transformation. Thus, it is heuristic to define $\boldsymbol{p}_c^i$ and $\boldsymbol{q}_c^i$ as follows:

$$\begin{cases} \boldsymbol{p}_c^i = \boldsymbol{p}_i; \quad \boldsymbol{q}_c^i = \boldsymbol{q}_i & \text{for each } j \in \mathcal{R}_i \text{ is not a constraint node,} \\ \boldsymbol{p}_c^i = \boldsymbol{p}_s; \quad \boldsymbol{q}_c^i = \boldsymbol{q}_s & \exists s \in \mathcal{R}_i \text{ is a constraint node.} \end{cases} \tag{7}$$

In our experiments, the deformations governed by the energy function in [26] may be inconsistent when the

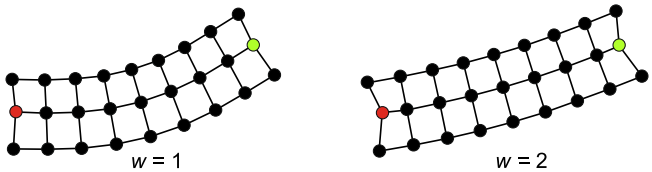

Fig. 4. Deformation of lattice in Fig. 2 under the energy function in [26]. The result is inconsistent when the global stiffness is increased

global stiffness increases, as illustrated in Fig. 4. However, the proposed energy function can effectively avoid such an inconsistency, which is illustrated at the bottom row of Fig. 2.

In the energy function Eq. 6, we carefully separate the node j and the region i, so that optimization computation can be speeded up, which is described in detail in Sect. 3.3.1.

3.3 Energy function optimization

Now, we describe the optimization method that minimizes the energy function Eq. 6 subject to the user's constraints.

Let us first discuss how to specify the constraints. In the 3D case, a user manipulates the model by selecting constraint primitives (vertices or faces) on the object. The specified constraints are automatically mapped to the lattice node constraints where the cells intersect the constraint faces or contain the constraint vertices. In the 2D case, a user can simply select the lattice nodes as the constraints.

After the constraints are specified, the algorithm computes the deformed positions of the other lattice nodes $\{\boldsymbol{q}_i\}$ by minimizing the energy $E(\mathcal{L})$. It is a nonlinear optimization problem since the rigid transformations $\{\boldsymbol{A}_i\}_{i=1}^n$ are dependent on the unknown variables $\{\boldsymbol{q}_i\}$ and cannot be linearly parameterized [26]. Like methods in [4, 26, 30], the problem can be solved by using an iterative Newton-type method. Starting from initial guesses of $\{\boldsymbol{q}_i\}$, the iterative process works as follows.

At the kth iteration, first the rigid transformations $\{\boldsymbol{A}_i\}^{(k)}$ are solved by minimizing the energy $E(\mathcal{L})$ where the node positions $\{\boldsymbol{q}_i\}^{(k-1)}$ are chosen as the values computed at the last iteration. Then, the new node positions $\{\boldsymbol{q}_i\}^{(k)}$ are computed by using the current rigid transformations $\{\boldsymbol{A}_i\}^{(k)}$ and the positions of constraints. The iteration process stops when the local energy minimization is reached. In our implementation, the initial guesses of $\{\boldsymbol{q}_i\}$ are taken as the deformed positions of lattice nodes at the last deformation step, which works well in our implementation.

Since each term in the energy sum $E(\mathcal{L})$ in Eq. 6 involves only the local region rigid transformation $\boldsymbol{A}_i$, the optimal rigid transformation for each local rigid region can be computed regardless of other local rigid regions and their optimal rigid transformations. Thus, each rigid transformation $\boldsymbol{A}_i$ can be computed by minimizing the local rigid region energy $E(\mathcal{R}_i)$ in Eq. 2. The solutions to $\boldsymbol{A}_i$ of the 2D and 3D cases are Eq. 3 and the rotational part of Eq. 5, respectively, which were described in Sect. 3.2.

With the specified constraints and the computed optimal rotations $\{\boldsymbol{A}_i\}$, the unknown lattice node positions $\{\boldsymbol{q}_i\}$ can be computed through Eq. 6 by solving a linear least square problem. After computing the gradients of $E(\mathcal{L})$ with respect to the unknown node positions $\{\boldsymbol{q}_i\}$ and setting the gradients to zero, the following sparse linear system of functions is obtained:

$$\begin{cases} \text{For each } j \in \mathcal{R}_i \text{ is not a constraint node} \\ \displaystyle\sum_{j\in\mathcal{R}_i} -w_i(\boldsymbol{q}_j-\boldsymbol{q}_i)+w_j\big(\boldsymbol{q}_i-\boldsymbol{q}_c^j\big) \\ \quad \displaystyle= \sum_{j\in\mathcal{R}_i} -w_i\boldsymbol{A}_i(\boldsymbol{p}_j-\boldsymbol{p}_i)+\sum_{j\in\mathcal{R}_i} w_j\boldsymbol{A}_j\big(\boldsymbol{p}_i-\boldsymbol{p}_c^j\big), \\ \exists s \in \mathcal{R}_i \text{ is a constraint node} \\ \displaystyle\sum_{j\in\mathcal{R}_i} w_j\big(\boldsymbol{q}_i-\boldsymbol{q}_c^j\big) = \sum_{j\in\mathcal{R}_i} w_j\boldsymbol{A}_j\big(\boldsymbol{p}_i-\boldsymbol{p}_c^j\big), \end{cases} \tag{8}$$

which can be rewritten in a compact matrix form:

$$\boldsymbol{M}\boldsymbol{q}=\boldsymbol{b}, \tag{9}$$

where $\boldsymbol{q}$ is a vector composed of all unknown nodes $\boldsymbol{q}_i$ and $\boldsymbol{b}$ is a vector whose ith entry is the right-hand side of the ith expression of the linear system Eq. 8. Note that the vector $\boldsymbol{b}$ should also be updated to incorporate the constraints into the system at each iteration.

3.3.1 Optimization implementation

At each iteration of the nonlinear optimization process, the sparse linear system Eq. 9 needs to be solved, and it is necessary to compute the coefficient matrix $\boldsymbol{M}$ and the vector $\boldsymbol{b}$. Note that the matrix $\boldsymbol{M}$ depends only on the initial local rigid region configuration. Thus, a direct solver with pre-factorization can be employed for minimizing $E(\mathcal{L})$, where the matrix $\boldsymbol{M}$ need to be factored only once [6]. In this way, the system Eq. 9 could be solved efficiently.

To compute the vector $\boldsymbol{b}$ in Eq. 9, the optimal rigid transforms $\{\boldsymbol{A}_i\}$ in Eq. 3 or the rotational part of Eq. 5 should be computed first. It is obvious that the computational costs of $\{\boldsymbol{A}_i\}$ and $\boldsymbol{b}$ depend on the size of the local rigid region, which increases quadratically in 2D (cubically in 3D) with the region half-width w. Thus, the naive calculation scheme will be slow for the stiff model. In our implementation the fast summation algorithm is adopted to address this problem [21]. The essence of the fast summation algorithm is to reuse the redundant summation computations on the lattice regions, and eventually reduce the runtime of calculations on each region to be constant and independent of the region half-width w. Here, we briefly introduce the fast summation operator and describe how to accelerate the computations of 3, 5 and the vector $\boldsymbol{b}$ by expressing them in terms of summations.

Using fast summation algorithm. The fast summation operator indicates the sum of an attribute v_j over $j \in \mathcal{R}_i$ [21]:

$$\mathcal{F}\{v_j\} \equiv \sum_{j\in\mathcal{R}_i} v_j \quad \text{(using fast summation)}.$$

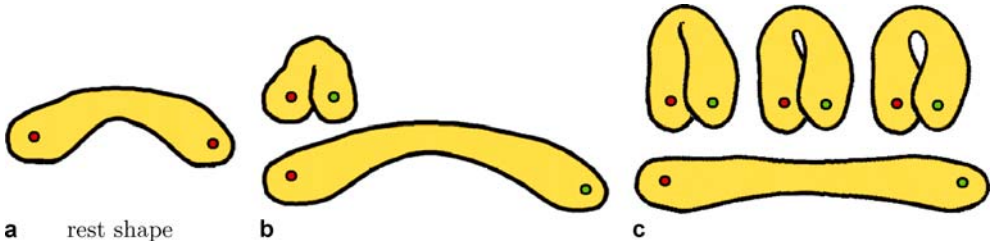

Fig. 5a–c. Comparison of our approach (**c**) with the method in [11] (**b**). The *upper row* results of (**c**) are with different global stiffnesses from soft to stiff ($w = 1$, 3 and 6), while the stretching result at the *lower row* is with the medium global stiffness ($w = 3$)

Fig. 6a–c. Comparison of our approach (**c**) with the nonlinear method in [30] (**b**). The latter produces shrinkages at the *right knee* and the *left elbow*

Note that the fast summation could not directly applicable to 3, 5 and the right-hand side of Eq. 8 because of the coupling of the node j and the region i. However, these dependences can be decoupled by simply expanding and restating the formula. For example, the term $\sum_{j\in\mathcal{R}_i} w_j \boldsymbol{A}_j(\boldsymbol{p}_i - \boldsymbol{p}_c^j)$ on the right-hand side of Eq. 8 can be expanded as follows:

$$\sum_{j\in\mathcal{R}_i} w_j \boldsymbol{A}_j\big(\boldsymbol{p}_i - \boldsymbol{p}_c^j\big) = \Big(\sum_{j\in\mathcal{R}_i} w_j \boldsymbol{A}_j\Big)\boldsymbol{p}_i - \sum_{j\in\mathcal{R}_i} w_j \boldsymbol{A}_j \boldsymbol{p}_c^j$$
$$= \mathcal{F}\{w_j \boldsymbol{A}_j\}\boldsymbol{p}_i - \mathcal{F}\big\{w_j \boldsymbol{A}_j \boldsymbol{p}_c^j\big\}. \quad (10)$$

Given w_j, $\boldsymbol{A}_j$ and $\boldsymbol{p}_c^j$ defined on each lattice node, the term Eq. 10 can be efficiently calculated by using the fast summation algorithm. Another term $\sum_{j\in\mathcal{R}_i} -w_i \boldsymbol{A}_i (\boldsymbol{p}_j - \boldsymbol{p}_i)$ on the right-hand side of Eq. 8 can be processed similarly, as may 3 and 5.

To further speed up the computation of $\{\boldsymbol{A}_i\}$ and $\boldsymbol{b}$, it will be helpful to pre-compute and reuse information as much as possible. For example, in the 2D case, $\sum_{j\in\mathcal{R}_i} p_j = \mathcal{F}\{\boldsymbol{p}_j\}$ can be pre-computed and used in all expanded terms of Eq. 3 at each iteration.

3.4 Stiffness tuning

In the proposed deformation, global and local stiffness controls are provided. By adjusting the global and local stiffness parameters, i.e. w and w_i, users can effectively control the deformation behaviors of a model, as illustrated in Figs. 1, 5c, 8 and 9. For the interaction consideration, an intuitive interface is designed to specify the global and local stiffness parameters.

The default value of $\{w_i\}_1^n$ is 1 for a homogeneous object. To mimic the inhomogeneous object deformation, $\{w_i\}_1^n$ should be specified differently. For the sake of simplicity, three levels (standard, enhanced, hard) of local stiffness are provided with the corresponding $w_i = \{1, (2w+1), (2w+1)^2\}$. An intuitive sketch interface is designed to assign the desired local stiffness. Our experiments show that this strategy provides enough flexibility to adjust the local stiffness. Of course, the levels could be extended and it is also reasonable to assign the desired w_i directly to the selected components.

The global stiffness can also be specified in a similar manner with three pre-defined levels, i.e. 'Soft', 'Medium'

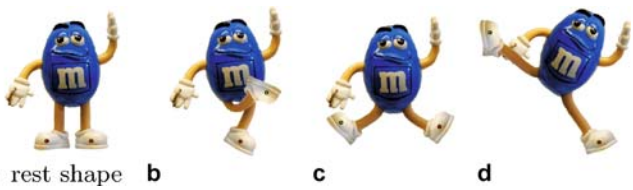

Fig. 7a–d. A cartoon motion sequence is generated by manipulating only two constraints at the foot

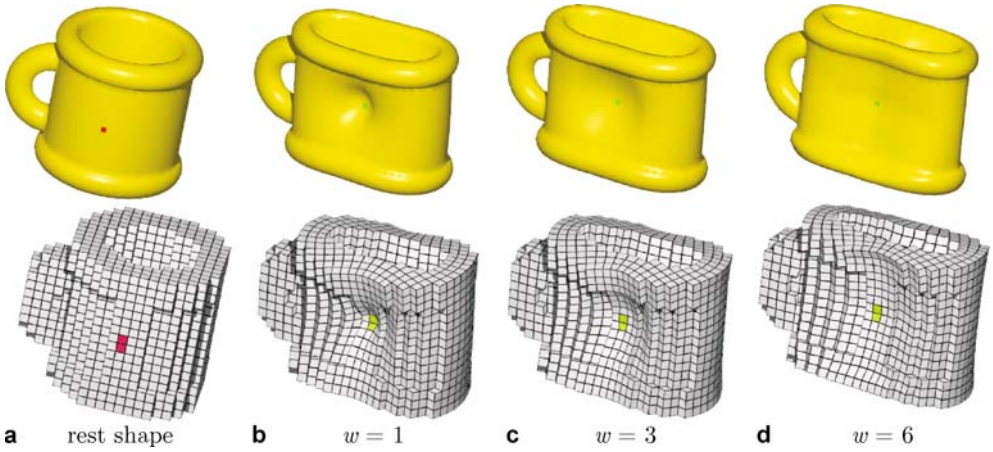

Fig. 8a–d. A cup is deformed with different global stiffnesses

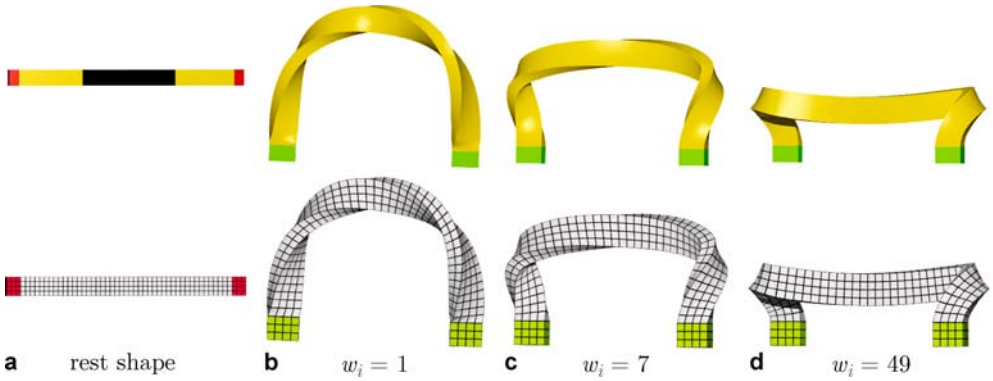

Fig. 9a–d. A bar is twisted and bent with different local stiffnesses at the middle part where the global stiffness $w = 3$

and 'Stiff' corresponding to $w = 1$, 3 and 6, respectively. Similar to the local stiffness control, the levels for global stiffness control can be extended and w can also be user specified.

4 Implementation and discussion

The method has been implemented and tested for 2D and 3D models on a PC with a 2.4 GHz Pentium 4 CPU and 2 GB memory. In our implementations, the nonlinear optimization will be convergent after 6 to 12 iterations in general. Table 1 shows the statistics of geometry data and runtime for the examples presented in the paper, where the optimization includes all iterations. The time for transforming the lattice deformation to the object is neglected because it is less than 1 ms for all examples in the paper. With the increase of the region width w, the number of non-zero elements in the sparse linear system Eq. 9 will also increase, which leads to the sparse linear system solver performance drop as well [6]. Due to employing the fast summation algorithm, the proposed deformation can still be performed interactively for not only soft models but also stiff models which are voxelized with a modest number of lattice nodes as shown in Table 1.

Figures 1 and 5–7 illustrate the 2D shape deformations. As shown in these examples, the nonlinear energy function for the overlapping local rigid regions can generate the natural and physically plausible deformations with a few constraints. For example, in Fig. 7, a natural motion sequence of the 2D character is achieved by manipulating only two lattice nodes at the feet. The result is comparable to that by the nonlinear method [30], as well as the linear method [11]. In some cases, our nonlinear method can produce more physically plausible results than [11], which is illustrated in Fig. 5. Meanwhile, the proposed method can manifest the shape rigidity well by enforcing the rigidity of the boundary and the interior of the shape. The method in [30] tries to preserve the local areas with the mean value coordinates and edge length constraints; it is not enough to keep the shape's interior rigidity and makes

Table 1. Performance statistics. VER NUM: number of vertices; NOD NUM: number of lattice nodes; VOX: runtime of initial voxelization; w: region half-width; PRE: runtime of pre-computation; OPTM: runtime of Newton-type iterations

Fig.	VER NUM	NOD NUM	VOX (ms)	w	PRE (ms)	OPTM (ms)
1	607	1415	0.05	1	12.5	9.3
				3	42.95	18.6
				6	109.2	32.7
7	537	1313	0.01	3	31.2	14.1
8	5668	3243	218	1	92.05	98.1
				3	697.35	248.1
				6	1634.9	276
9	2946	780	17	3	24.95	23.4
10	35 448	1224	437	1	14.85	37.5

the deformation results shrunken sometimes. By using the shape which appears in Fig. 8 of [30], we compare the deformations between our method and [30] in Fig. 6.

Since our method provides the intuitive local and global stiffness controls, the deformation is more flexible. Figures 1 and 5c show the deformations with the different global stiffness parameters from soft to stiff under the same constraints. In Fig. 1e, the local stiffness has been adjusted to manifest the anisotropic material of the object.

The method has also been applied to 3D models. By observing the examples in Figs. 8–10, the deformation results generated by our algorithm are comparable to those generated by the state-of-the-art nonlinear [10] or linear methods [27]. Figure 8 shows the deformations of a cup model with different global stiffness parameters from soft to stiff under the same constraints conducted on the cup sides. We can see that the deformation results in Fig. 8b–d are similar to the deformations of the cups made of mud, sponge and iron, respectively. Another example is shown in Fig. 9, which illustrates the deformation results of different local stiffnesses at the middle part of a bar under the same twist and bend constraints. By enhancing the local stiffness, the twisting and bending at the middle part has been effectively alleviated. Figure 10 shows the shape editing on a trianglar soup model. The 3D character model is converted from a triangular mesh and it contains some deliberately produced self-intersections and degenerated triangles. Since the proposed algorithm decouples the deformation from the underlying geometry representation, it can handle the disconnected components and self-intersections easily.

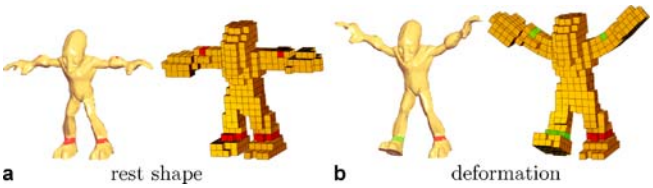

Fig. 10a,b. A triangular soup model is deformed with the global stiffness $w = 1$

5 Conclusion and future work

The paper proposed a shape deformation algorithm that incorporates global and local stiffness controls. The algorithm voxelizes the object as a regular lattice, and organizes the lattice cells as overlapping local rigid regions. During deformation, a nonlinear energy function is optimized to minimize the distortion of the overlapping local rigid regions, which yields physically plausible deformations. Combining the fast summation technique, the interactive deformation is achieved for not only soft but also stiff objects.

In our setting, the region width determines the global stiffness and a penalty coefficient for each local rigid region controls the local stiffness; thus, the user can adjust the global and local stiffnesses conveniently and intuitively for the desired deformation behaviors.

There are still improvements for the proposed method. Currently, the user can increase (decrease) the region width to increase (decrease) the global stiffness. However, sometimes it is desirable to decide the region width automatically so that the global stiffness corresponds to a specified material in the real world, e.g. wood, iron or steel. Meanwhile, it is possible to learn the local stiffness from given examples using the technique in [20]. In addition, we are considering using an adaptive lattice to approximate the object to improve the algorithm efficiency further.

Acknowledgement The authors would like to thank Dr. Kun Zhou for providing the software and the 2D shape in Fig. 6, and Dr. Takeo Igarashi for the 2D shape in Fig. 7. The work is jointly supported by the NSF of China (Nos. 60743002 and 60736019), the NSF of Zhejiang Province (No. R106449), the Doctoral Program of Higher Education (Specialized Research Fund) (No. 20050335069) and the China Postdoctoral Science Foundation (No. 20070421184).

References

1. Au, O.K.C., Fu, H., Tai, C.L., Cohen-Or, D.: Handle-aware isolines for scalable shape editing. ACM Trans. Graph. (SIGGRAPH 2007) **26**(3), 83 (2007)
2. Au, O.K.C., Tai, C.L., Liu, L., Fu, H.: Dual Laplacian editing for meshes. IEEE Trans. Vis. Comput. Graph. **12**(3), 386–395 (2006)
3. Botsch, M., Pauly, M., Gross, M., Kobbelt, L.: PriMo: coupled prisms for intuitive surface modeling. In: SGP '06: Proceedings of the Fourth Eurographics

Symposium on Geometry Processing. Eurographics Association, Cagliari, Sardinia, Italy (2006)
4. Botsch, M., Pauly, M., Wicke, M., Gross, M.: Adaptive space deformations based on rigid cells. Comput. Graph. Forum (Eurographics 2007) **26**(3), 339–347 (2007)
5. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. **14**(1), 213–230 (2008)
6. Davis, T.A.: UMFPACK: unsymmetric multifrontal sparse LU factorization package. http://www.cise.ufl.edu/research/sparse/umfpack/. Visit on 10.12.2007
7. Gibson, S., Mirtich, B.: A survey of deformable modeling in computer graphics. Tech. Rep. No. TR-97-19, Mitsubishi Electric Research Lab., Cambridge (1997)
8. Güdükbay, U., Özgüç, B., Tokad, Y.: A spring force formulation for elastically deformable models. Comput. Graph. **21**(3), 335–346 (1997)
9. Horn, B.K.P.: Closed-form solution of absolute orientation using unit quaternions. J. Opt. Soc. Am. A **4**, 629–642 (1987)
10. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. ACM Trans. Graph. (SIGGRAPH 2006) **25**(3), 1126–1134 (2006)
11. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Graph. (SIGGRAPH 2005) **24**(3), 1134–1141 (2005)
12. James, D.L., Barbič, J., Twigg, C.D.: Squashing cubes: automating deformable model construction for graphics. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches, p. 38. ACM Press, Los Angeles, CA (2004)
13. James, D.L., Pai, D.K.: ArtDefo: accurate real time deformable objects. In: SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 65–72. ACM Press, Los Angeles, CA (1999)
14. Kraevoy, V., Sheffer, A.: Mean-value geometry encoding. Int. J. Shape Model. **12**(1), 29–46 (2006)
15. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: Akeley, K. (ed.) SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 165–172. ACM Press, New Orleans, LA (2000)
16. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. (SIGGRAPH 2005) **24**(3), 479–487 (2005)
17. Müller, M., Heidelberger, B., Teschner, M., Gross, M.: Meshless deformations based on shape matching. ACM Trans. Graph. (SIGGRAPH 2005) **24**(3), 471–478 (2005)
18. Nealen, A., Müller, M., Keiser, R., Boxerman, E., Carlson, M.: Physically based deformable models in computer graphics. In: Eurographics: State of the Art Report. Eurographics Association, Dublin, Ireland (2005)
19. Platt, J.C., Barr, A.H.: Constraints methods for flexible models. ACM Comput. Graph. (SIGGRAPH 1988) **22**(4), 279–288 (1988)
20. Popa, T., Julius, D., Sheffer, A.: Material-aware mesh deformations. In: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006, p. 22. IEEE Computer Society, Matsushima, Japan (2006)
21. Rivers, A.R., James, D.L.: Fastlsm: fast lattice shape matching for robust real-time deformation. ACM Trans. Graph. (SIGGRAPH 2007) **26**(3), 82 (2007)
22. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Trans. Graph. (SIGGRAPH 2006) **25**(3), 533–540 (2006)
23. Schiwietz, T., Georgii, J., Westermann, R.: Freeform image. In: Proceedings of Pacific Graphics 2007, pp. 27–36. IEEE Computer Society, Maui, Hawaii (2007)
24. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. ACM Comput. Graph. (SIGGRAPH 1986) **20**(4), 151–160 (1986)
25. Shoemake, K., Duff, T.: Matrix animation and polar decomposition. In: Booth, K.S., Fournier, A. (eds.) Proceedings of the Conference on Graphics Interface '92, pp. 258–264. Morgan Kaufmann Publishers Inc., Vancouver, British Columbia (1992)
26. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: SGP '07: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, pp. 109–116. Eurographics Association, Barcelona, Spain (2007)
27. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. ACM Press, Nice, France (2004)
28. Sumner, R.W., Schmid, J., Pauly, M.: Embedded deformation for shape manipulation. ACM Trans. Graph. (SIGGRAPH 2007) **26**(3), 80 (2007)
29. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. ACM Comput. Graph. (SIGGRAPH 1987) **21**(4), 205–214 (1987)
30. Weng, Y., Xu, W., Wu, Y., Zhou, K., Guo, B.: 2d shape deformation using nonlinear least squares optimization. Visual Comput. **22**(9), 653–660 (2006)
31. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with Poisson-based gradient field manipulation. ACM Trans. Graph. (SIGGRAPH 2004) **23**(3), 644–651 (2004)
32. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph Laplacian. ACM Trans. Graph. (SIGGRAPH 2005) **24**(3), 496–503 (2005)

WENWU YANG is a Ph.D. candidate in the State Key Lab of CAD&CG, Zhejiang University, Peoples' Republic of China. He received his M.Sc. degree in computer graphics from Zhejiang University in 2005. His research interests include computer graphics, geometric modeling and computer animation.

JIEQING FENG is a professor in the State Key Lab of CAD&CG, Zhejiang University, Peoples' Republic of China. He received his B.Sc. degree in applied mathematics from the National University of Defense Technology in 1992 and his Ph.D. degree in computer graphics from Zhejiang University in 1997. His research interests include geometric modeling, rendering and computer animation.

XIAOGANG JIN is a professor in the State Key Lab of CAD&CG, Zhejiang University, Peoples' Republic of China. He received his B.Sc. degree in computer science in 1989 and M.Sc. and Ph.D. degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His current research interests include implicit surface computing, special effects simulation, mesh fusion, texture synthesis, crowd animation, cloth animation and non-photorealistic rendering.