2D Shape Deformation Based on Rigid Square Matching

A. Author
Author's Affiliation
Author's E-Mail
Author's Website

B. Author
Author's Affiliation
Author's E-Mail
Author's Website

C. Author
Author's Affiliation
Author's E-Mail
Author's Website

Abstract

In this paper, we propose a fast and stable method for 2D shape deformation based on rigid square matching. Our method utilizes uniform quadrangular control meshes for 2D shapes and tries to maintain the rigidity of each square in the control mesh during user manipulation. A rigid shape matching method is performed to find an optimal pure rotational transformation for each square in the control mesh. An iterative solver is proposed to compute the final deformation result for the entire control mesh by minimizing the difference between the deformed vertices and their counterparts in the neighboring rigid square. The deformation result on the 2D shape is as rigid as possible and the details of the shape are preserved well. As extensions, we present a shape-aware splitting method to improve the deformation effect for coarse meshes and a simple sketch-based clustering method for skeletal deformation. Experiments with various 2D shapes show that our method is efficient and easy to use, and can provide physically plausible result for shapes of objects in real world. Therefore, our shape deformation method is especially suitable for applications in cartoon character animation.

Keywords: shape deformation, character animation, shape matching, rigid transformation, skeletal deformation

Figure 1: Horsemanship actions obtained by our method. The original 2D shape with its control mesh is on the top left. Note the natural skeletal behavior in deforming the horse legs.

1. Introduction

2D shape deformation (or manipulation) aims at deforming objects represented in the form of 2D images by using graphical method. It is a very useful tool for applications such as character animation [1], real-time live performance [2] and enriching graphical interface [3], and therefore, has received a lot of attentions in recent years. In such typical applications, the 2D shape in the image to be deformed often represents an object in real world, and more often a character or an animal. As a result, the deformed shape must exhibit deformation effects that are physically plausible. Efficiency

is another important feature to be considered of the deformation method, because the method must provide the user interactive tools to manipulate the deformation result. The interaction metaphor for the user should also be intuitive and easy to use. Many methods have been proposed to satisfy these requirements.

Shape deformation methods using space-based techniques deform the shapes by manipulate the space in which they are embedded. Free-form deformation (FFD) methods [4, 5, 6] and skeleton-based techniques [7, 8, 9] are of this category. They are very efficient in computation and easy to be implemented. However, they don't provide convenient or meaningful interaction tools for the user. For FFD methods, the control is not intuitive, and for skeleton-based techniques, the weight tuning for rigging is a painful process for users.

Physically-based simulation including finite element method [10] and mass-spring system [11] can provide precise deformation result. But they are often too expensive in computation or too slow to converge for interactive applications. Moreover, precision of the deformation result is not very important in such applications as compared to efficiency and usability. Therefore, almost all the shape deformation methods proposed in recent years utilize geometrically-based approach instead of physically-based simulation.

Recently, many nonlinear mesh deformation methods [12, 13, 14, 15] are proposed for 3D models. They try to minimize a nonlinear energy functional representing local properties of the surface. Weng et al. [1] use a variant of these methods for 2D shape deformation. They use a hybrid mesh generated by using the shape boundary as the deformable model. Their method not only preserves the Laplacian coordinates like the deformation methods for 3D meshes do, but also preserves local area of the shape interior. All these terms add up into a non-quadratic energy function which is minimized by using an iterative Gauss-Newton method.

1.1 Related Work

2D shape deformation is closely related to a more general technique named image deformation, which focuses on reasonably warping the entire space of the image. Image deformation has a longer history and is typically used for 2D morphing [16] and medical imaging [17]. Schaefer et al. [18] proposed an image deformation method which can also be used for 2D shape deformation. They used moving least squares to maintain the rigidity of the entire image space. Though it is very efficient, the method doesn't take the information of the shape into account. Therefore, the details of the shape cannot be preserved well for large scale deformation. We instead utilize the explicit 2D rotation expression in [18] to maintain the local rigidity of the shape.

Müller et al. [19] proposed a meshless deformation method based on shape matching. Their method is fast and stable, and can produce physically plausible deformation results. Because of the simple modes of deformation (only linear and quadratic) used in shape matching, the method is only suitable for modest deformation complexity. A practical enhancement is proposed in [19] using overlapping domains for many shape matching clusters to enrich the deformation effect, but the result still have blending artifacts. Rivers and James [20] proposed a fast lattice shape matching (FastLSM) method to address these problems. Our method also utilizes regular lattices and shape matching technique, but there are also many significant differences. Our method is applied for 2D shape deformation aiming at applications in character animation and live performance. Their method overlaps many cells and therefore need a special fast algorithm to get the final result while our method only perform shape matching for each square independently. FastLSM uses region-based convolution to get the final result, while our method uses an iterative solver.

Igarashi et al. proposed the concept of as-rigid-as-possible manipulation in [21]. In their work, the 2D shape boundary is firstly represented by a 2D simple polygon and is then triangulated to form a triangular mesh. During manipulation, the user drags some vertices of the triangular mesh as handles, and the system computes the position of other free vertices by minimizing the deformation distortion of every triangle.

1.2 Our Contributions

In this paper, we propose a fast and stable method for 2D shape editing. It is motivated by the shape matching method for dynamic deformation in [19] and is very simple to be implemented. Because our shape editing me-

thod is mainly designed for applications in character animation, we also adopt the concept of as-rigid-as-possible manipulation in [21] to give physically plausible results. Our contributions can be summarized as follows:

(1) We propose a rigid square matching method for 2D shape deformation. This method is especially suitable for applications in character animation, because it tries to maintain the local rigidity of the 2D shape and can produce physically plausible de-formation effects for shapes of objects in the real world. To solve for the entire control mesh, we propose an iterative solver which utilizes the rigid transformations acquired by rigid square matching. The method is unconditionally stable, easy-implemented and efficient for interactive 2D shape editing.

(2) As a practical enhancement, we propose a shape-aware splitting method for the uniform quadrangular lattice. It exploits the topological information of the 2D shape to improve the deformation effect under a relatively coarser mesh;

(3) We implement skeletal deformations by using a user defined clustering method. The sketch-based metaphor is very convenient to use and the method is easy to implement.

2. Overview

In this section, we give an overview of our method which is shown in Figure 2. Our 2D shape deformation method takes an image as the input. We suppose that the 2D shape to be considered has salient differences in illumination or color from the background in the image. We also suppose that all the images are stored in the bitmap format for convenience of discussion, and the method proposed in this paper can be generalized to other image format easily. Firstly, the background which we are not interested in is removed manually. Different from some mesh-based methods [1, 21], our method doesn't need the shape boundary information. Therefore, the next step isn't the boundary extraction operation. Instead, the 2D shape is directly embedded into a regular quadrangular lattice. Then, by eliminating the lattice vertices outside the shape, we can obtain a regular quadrangular lattice tightly bounding the 2D shape. A uniform quadrangular mesh $\mathfrak{M} = (V, S)$ is generated with no difficulty from the lattice and is then used as the deformation control mesh, where V is the set of n vertices in the mesh and S is the set of m square cells (because we try to preserve their original shape during the deformation, we always call them "squares" in this paper despite they are not real squares any more after being deformed). Our method performs some pre-processing for this control mesh beforehand to decrease the real-time computation. Then the user can drag the mesh vertices freely to deform the 2D shape. During the deformation process, our method tries to maintain the local rigidity of each square to reserve the local details of the 2D shape. After the control mesh is deformed according to our method, the final 2D shape is rendered by using simple linear texture mapping technique for each square.

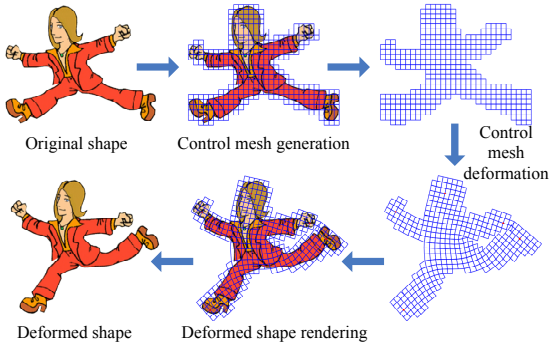


Figure 2: Overview of our 2D shape deformation method.

The remainder of this paper is organized as follows. Section 3 describes the core part of our method in details, including the rigid square matching algorithm and the iterative solver to calculate the deformation result of the entire mesh. In Section 4, two extensions for our method are proposed to improve the deformation result and enrich the deformation style. Experimental results are shown in Section 5, followed by conclusions in Section 6.

3. Shape Deformation Based on Rigid Square Matching

After the control mesh generation phase, a uniform quadrangular mesh $\mathfrak{M} = (V, S)$ is generated. Then the user can drag any vertices in this mesh to deform the 2D shape. As depicted in Figure 2, our method firstly deforms the control mesh according to the "as-

rigid-as-possible" principle. The rigid square matching method is proposed to maintain the rigidity of each square in the control mesh. We use a simple iterative solver to compute the deformation result for the entire control mesh.

3.1 Rigid Square Matching

The progress of the rigid square matching algorithm is illustrated in Figure 3. It tries to find a "rigid square" which fits the current deformed square best. Firstly, we consider an individual square $s \in S$ in the control mesh. The original positions of its four vertices are $\mathbf{x}_i^0 \in \mathbb{R}^2$ $(1 \le i \le 4)$. After the user's manipulation, the new positions of these vertices are $\mathbf{x}_i \in \mathbb{R}^2$. According to the shape matching technique used in [19], the optimal rigid transformation, including an optimal 2D rotation $\mathbf{R} \in \mathbb{R}^{2\times 2}$ and a translation vector $\mathbf{t} \in \mathbb{R}^2$, are defined to minimize the difference between transformed $\mathbf{x}_i^0$ and $\mathbf{x}_i$

$$\sum_{i=1}^{4} \omega_i \left| \mathbf{R}\mathbf{x}_i^0 + \mathbf{t} - \mathbf{x}_i \right|^2 \qquad (1)$$

where ω_i $(1 \le i \le 4)$ are weights of individual vertices. While Müller et al. [19] use the mass of each point as its weight because they aim at real-time dynamic deformation, we use the weighting scheme to implement position constraint for handles. In all the examples presented in this paper, we find that assigning $2n$ for constrained handles and 1 for other free vertices is a good choice.

We can actually remove the translation vector $\mathbf{t}$ in Equation 1 to simplify the minimization problem. Setting the partial derivatives with respect to $\mathbf{t}$ in Equation 1 to zero yields the optimal translation vector

$$\mathbf{t} = \mathbf{x}_c - \mathbf{R}\mathbf{x}_c^0 \qquad (2)$$

where $\mathbf{x}_c^0$ and $\mathbf{x}_c$ are weighted centroids of the square before and after deformation respectively:

$$\mathbf{x}_c^0 = \frac{\sum_i \omega_i \mathbf{x}_i^0}{\sum_i \omega_i}, \qquad \mathbf{x}_c = \frac{\sum_i \omega_i \mathbf{x}_i}{\sum_i \omega_i} \qquad (3)$$

Then we can substitute Equation 2 into Equation 1 to get a simpler formula with only $\mathbf{R}$ as the unknown

$$\sum_{i=1}^{4} \omega_i \left| \mathbf{R}\hat{\mathbf{x}}_i^0 - \hat{\mathbf{x}}_i \right|^2 \qquad (4)$$

where $\hat{\mathbf{x}}_i^0 = \mathbf{x}_i^0 - \mathbf{x}_c^0$ and $\hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_c$.

Since the rotational transformation in 3D space is nonlinear, Müller et al. [19] firstly relax the problem to find an optimal linear transformation and then extract the rotational part from it. We don't need to do this because of the linearity of 2D rotation, so we can directly solve for $\mathbf{R}$. As a 2D rotation matrix, $\mathbf{R}$ is an orthogonal matrix, i.e. $\mathbf{R}^T\mathbf{R} = \mathbf{I}$. If $\mathbf{R}$ is represented in the form of a block matrix

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_1 & \mathbf{R}_2 \end{pmatrix}$$

where $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{R}^2$, then $\mathbf{R}_1\mathbf{R}_2 = 0$, and $\mathbf{R}_1^T\mathbf{R}_1 = \mathbf{R}_2^T\mathbf{R}_2 = 1$. By using this property and minimizing Equation 4, the optimal rotation matrix can be given as

$$\mathbf{R} = \frac{1}{\mu} \sum_{i=1}^{4} \omega_i \begin{pmatrix} \hat{\mathbf{x}}_i^{0T} \\ \hat{\mathbf{x}}_i^{0\perp T} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_i & \hat{\mathbf{x}}_i^{\perp T} \end{pmatrix} \qquad (5)$$

where

$$\mu = \sqrt{\left(\sum_{i=1}^{4} \omega_i \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_i^0 \right)^2 + \left(\sum_{i=1}^{4} \omega_i \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_i^{0\perp} \right)^2} \qquad (6)$$

and $\perp$ is an operator on 2D vectors such that $(x, y)^\perp = (-y, x)$.

Then, as the result of the rigid square matching method, the positions of the four vertices of the fitted rigid square are given by using this optimal rotation and translation

$$\tilde{\mathbf{x}}_i = \mathbf{R}\hat{\mathbf{x}}_i^0 + \mathbf{x}_c \quad (1 \le i \le 4) \qquad (7)$$

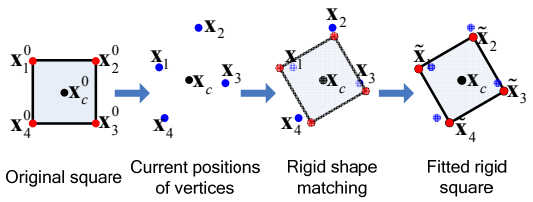


Figure 3: Progress of rigid square matching algorithm.

3.2 Iterative Solver for Control Mesh Deformation

By using the rigid square matching method proposed in Section 3.1, we can find a fitted rigid square for each deformed square. However, for the entire mesh, these rigid squares are not necessary to conform to their neighbors due to the arbitrary manipulation of the user. Thus, we propose an iterative solver to compute the new positions of the vertices in the entire control mesh by minimizing the difference between the resulting vertices in the deformed control mesh and their counterparts in the neighboring fitted rigid squares. For i th $(1 \le i \le n)$ vertex in the control mesh, its

current position is $\mathbf{x}_i$ and we suppose the set of its neighboring squares to be $N(i) \subseteq S$. Then, the error function for i th vertex is defined by

$$\sum_{s \in N(i)} \omega_i \left| \mathbf{x}_i - (\mathbf{R}_s \mathbf{x}_i^0 + \mathbf{t}_s) \right|^2 \qquad (8)$$

where $\mathbf{R}_s$ and $\mathbf{t}_s$ are the optimal rotation matrix and translation vector computed for the neighboring square s, respectively. Then the global error function over the entire control mesh is given by

$$\sum_i \sum_{s \in N(i)} \omega_i \left| \mathbf{x}_i - (\mathbf{R}_s \mathbf{x}_i^0 + \mathbf{t}_s) \right|^2 \qquad (9)$$

We need to minimize this error function to find the new positions for each vertex in the deformed control mesh. However, $\mathbf{R}_s$ and $\mathbf{t}_s$ are dependent in the positions of other vertices in the control mesh, so Equation 9 is non-quadratic and consequently, it cannot be solved directly. Therefore, we propose a simple iterative solver to linearize it. $\mathbf{R}_s$ and $\mathbf{t}_s$ are supposed to be invariant for each iteration and they can be viewed as constant in Equation 9. Then this quadratic function has a unique minimizer, which yields the iterative solver for the final positions of all the vertices in the control mesh

$$\mathbf{x}_i \leftarrow \frac{1}{\left\| N(i) \right\|} \sum_{s \in N(i)} (\mathbf{R}_s \mathbf{x}_i^0 + \mathbf{t}_s) \qquad (10)$$

For each iteration step, the fitted rigid squares for all the deformed squares are computed independently and then the corresponding rotations and translations are used to update the positions of all the vertices by using Equation 10. The iterative solver repeats until the total error function of the entire mesh expressed by Equation 9 varies less than a given threshold in several successive iterations. The experimental results show that the total error function becomes stable in tens of iterations after user manipulation (Figure 4). Since all the computation performed in each iteration is independent for each square or vertex and only deal with 2D matrix and vectors, the time complexity for one iteration is $O(m)$. As a result, our deformation method can be very efficient even for control meshes with very fine resolution.

Moreover, we make a little modification for the iterative solver in practice to further improve its interactive performance. When the user drags the handles, we perform the iteration for a fixed number of times (we use 30 times in all the examples) to guarantee the real-time response, because the intermediate results during user interaction need not to be precise. When the user releases the handles and finishes the interaction, our solver proceeds to find the precise result at convergence.

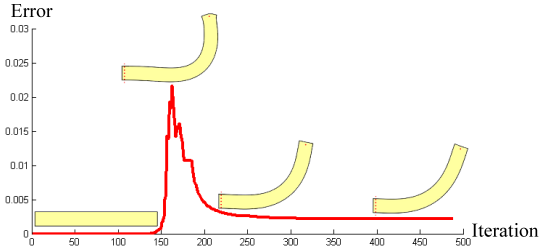


Figure 4: Error/iteration curve. The peak of the curve occurs when the user drags and releases the handle.

4. Extensions

4.1 Shape-aware Mesh Splitting

In order to simplify the control mesh generation, we use uniform quadrangular control meshes for shape deformation instead of adaptive meshes. As depicted in Figure 5, for some particular shapes with two geodesically remote parts located near in space, coarse meshes will introduce incorrect connections between the two parts. This problem is especially obvious for shapes like animals and humans which have obvious elongated parts such as limbs. One directly approach to solve this problem is to use a finer lattice for the shape to produce a finer control mesh, but this will increase the computation cost of the deformation algorithm. Instead, we propose a shape-aware mesh splitting method to alleviate this problem for relatively coarser mesh.

The main idea of the shape-aware mesh splitting method is to split parts of the mesh by duplicating vertices in order to break the incorrect connection in the control mesh. To achieve this, we further exploit the information of the shape, defining the "boundary vertices" to be the mesh vertices located outside the shape. Ideally, all the boundary vertices of a shape will form one or more closed loops, i.e. each boundary vertex has exactly two neighboring boundary vertices. But for incorrect connections in the control mesh, one boundary vertex (which is called the "invalid boundary vertex") may have more than two neighboring boundary vertices as depicted in Figure 5. For

such cases, the invalid boundary vertex is duplicated and assigned to the neighboring squares according to the 2D shape information by using a series of rules which is depicted in Table 1. In this table, the shaded part in each square represents the inner region of the shape. The red points are invalid boundary vertices to be considered and the blue points are duplicated boundary vertices which become valid. The duplicated vertices are offset a little for clarity. It should be noticed that new invalid boundary vertices may be introduced when one invalid boundary vertex is resolved according to the rules. Actually, the rules are applied to the control mesh repeatedly until there's no invalid boundary vertex. As a result, the incorrect connections of the control mesh are split and more natural deformation effect can be produced even for a relatively coarser mesh (Figure 7).

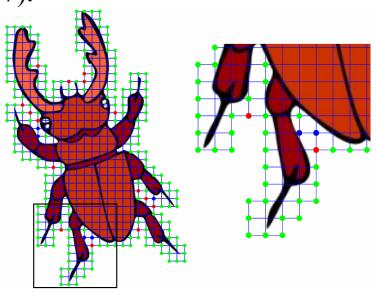


Figure 5: An example of incorrect connections. The zoomed depiction of the part enclosed by the black rectangle is shown on the right. All the colored points represent the boundary vertices. Red and blue points represent incorrect connections to be split. Initially, red points are recognized as invalid boundary vertices by our mesh splitting method.

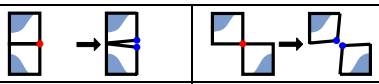
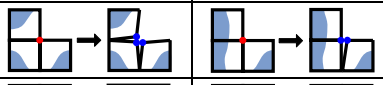
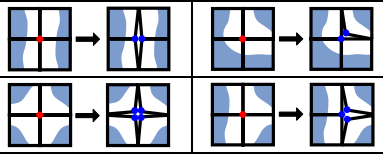
Number of neighboring squares	Duplication and assigning rules
2	
3	
4	

Table 1: Mesh splitting rules.

4.2 Skeletal Deformation Using Sketch-based Clustering

Skeletal deformation is very important for applications in character animation, because almost all cartoon characters are articulated animals. Therefore, we propose a skeletal deformation approach to our basic 2D shape deformation method. This approach is implemented by using clustering technique and provides a simple sketching metaphor to the user. We view each individual square in the control mesh as a deformation cluster and generate larger clusters which are composed of squares according to the user interaction to represent the skeletal structure of the shape. The entire progress of this approach is shown in Figure 6. The user draws lines on the shape to designate the skeletons. Then, the squares which intersect the same line are assembled into one cluster. When the user deforms the skeletal shape, each newly generated cluster is treated as a square in the above-mentioned rigid square matching algorithm and replaces the squares which make up it. Because our 2D shape deformation method tries to maintain the rigidity of every cluster, the skeletal deformation effect can be presented by using this simple sketch-based clustering approach.

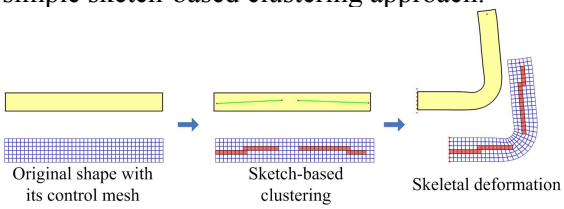


Figure 6: Skeletal deformation using sketch-based clustering.

5. Experimental Results

We have implemented the described 2D shape deformation method on a workstation with a 2.33GHz Intel® Core™2 Duo CPU and 2GB memory. Table 2 shows the statistics for the examples used in this paper and timings for our shape deformation method. In Table 2, "Lattice grid" means the resolution of the space lattice in which the shapes are embedded to generate the control meshes; "Solution time 1" means time need for our method to perform 30 iterations when the user drags the handles as described in Section 3.2; and "Solution time 2" means time need to exactly perform the iterative solver until it satisfies the stop condition to

get the final result when the user releases the handles. “Solution time 1” is approximately linear in the scale of the control mesh, indicating that our 2D shape deformation method is very efficient for interactive shape editing. “Solution time 2” is affected not only by the scale of the mesh, but also the scale of the deformation and the topology of the mesh, so the numbers listed in the last row of Table 1 are average time under various conditions for each shape. In all the experiments presented in this paper, the iterative solver converges within 0.6 second. This cost is acceptable because it only occurs when the user stops interacting with the 2D shape.

Figure 7 compares the deformation results with and without shape-aware mesh splitting for the same shape under the same resolution of control mesh. It is shown that the shape-aware mesh splitting method can eliminate the incorrect connections in the control mesh. By using the split control mesh, our 2D deformation method can provide more natural effect. In Figure 8, we present the effect of skeletal deformation by using our sketch-base clustering approach. It is also compared with the deformation result without skeletons. Our skeletal deformation approach provides an easy-to-use metaphor for users and produces physically plausible effect especially for articulated cartoon characters. Figure 9 shows that our method can provide more realistic result than the as-rigid-as-possible shape manipulation method in [21] does. To illustrate the versatility of our 2D shape deformation method, more examples are given in Figures 1 and 10. The timings in Table 2 and examples in the figures show that our method can provide as good deformation result as the nonlinear optimization method in [1] at less computation cost. Figures 11 and 12 show that our method can also be applied to image deformation to provide similar results as [18] with no extra modification except treating the entire image as a shape.

6. Conclusion

In this paper, we propose a 2D shape deformation method based on rigid square matching. The method uses uniform quadrangular meshes as control meshes which is much easier to build than triangular meshes used in [21] and hybrid meshes used in [1]. We adopt the concept of as-rigid-as-possible deformation and use the shape matching technique to maintain the local rigidity of the shape. The transformations acquired by shape matching technique are constrained explicitly to be pure rotations. Therefore, the details of the 2D shapes can be preserved well during deformation. Moreover, we directly update every vertices of one square by using pure rotational transformation without dynamic issues in [19], and consequently there will be no inverted square and our system is unconditionally stable. We also propose a simple iterative solver to compute the final deformation result of the entire mesh. The deformation mode for each square is simply the rigid motion, but the final results of the entire shape exhibit very complex deformation effects as shown in Figures 1 and 10, because the connections between the squares in the control mesh provide the system a very high degree-of-freedom. Essentially, our shape deformation method is space-based, but by generating the control mesh according to the 2D shape it avoid the problems presented in the space warping approaches. We also propose a practical enhancement which further utilizes the information of the shape to improve the deformation effect for coarse control meshes. By using a simple clustering method, our method can provide skeletal deformation effect for articulated shapes like cartoon characters and animals. The skeleton designating metaphor is sketch-based and is very easy to use.

Our method is very efficient and can provide physically plausible deformation effect for shapes of objects in real world. However, it still has some limitations. First, because we use pure rotational transformation for each square in the control mesh, the global area cannot be preserved by the current algorithm. Second, it is too rigid to deform shapes of soft and rubber-like objects, such as sponges and jellies. These two problems can be addressed by using more complex transformations in the square matching process. Moreover, since we are planning to generalize our method to 3D shape editing, a solver more efficient than the iterative solver proposed in Section 3.2 is to be considered in the future work.

2D shapes		Bee (Fig. 8, 10)	Horse (Fig. 1)	Gecko (Fig. 6)	Character (Fig. 2, 10)	Flower (Fig. 10)	Mona Lisa (Fig. 11)	Leaning Tower (Fig. 12)
Lattice grid		60×60	70×70	31×31	40×40	80×80	60×60	60×60
Control mesh statistics	n	1145	1965	584	718	2131	3233	3477
	m	975	1714	434	602	1913	3102	3360
Solution time 1		15.7ms	28.0ms	4.6ms	9.5ms	29.5ms	46.9ms	50.1ms
Solution time 2		0.39s	0.56s	0.14s	0.19s	0.59s	0.30s	0.51s

Table 2: Statistics and timings.

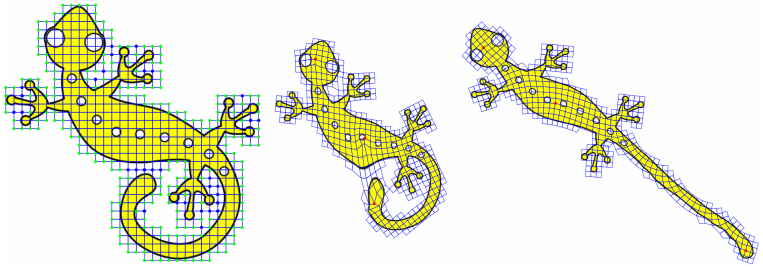


Figure 7: Deformation of a gecko with (right) and without (middle) shape-aware mesh splitting. The original shape of the gecko with the original control mesh is shown on the left. Boundary vertices are marked as green points and blue points with blue ones indicating incorrect connections before mesh splitting.

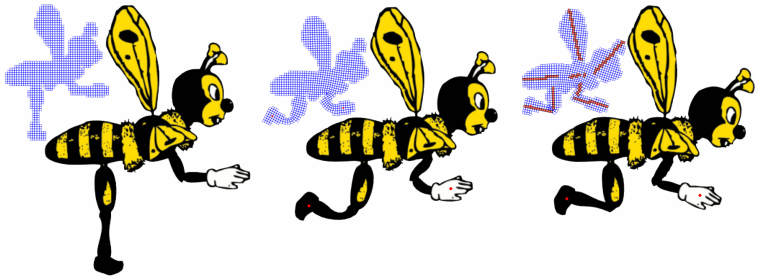


Figure 8: Deformation of a bee with (right) and without (middle) skeleton designation by sketch-based clustering. The original shape is shown on the left and the corresponding control meshes are shown beside the shapes. Clustered squares are painted in red for the control mesh of the deformed shape on the right. Red points represent the handles manipulated by the user.

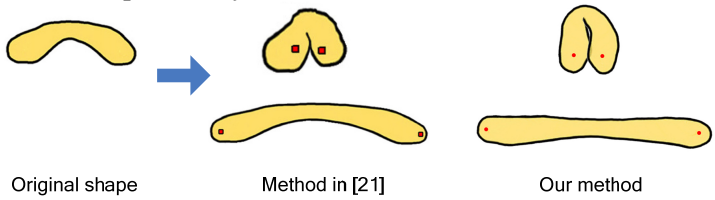


Figure 9: Comparison between the method in [21] and our method.

Acknowledgements

References

[1] Y. Weng, W. Xu, Y. Wu, K. Zhou, and B. Guo. 2D shape deformation using nonlinear least squares optimization. The Visual Computer, 22(9-11):653-660, 2006

[2] T. Ngo, D. Cutrell, J. Dana, B. Donald, L. Loeb, and S. Zhu. Accessible animation and customizable graphics via simplicial configuration modeling. In Proceedings of ACM SIGGRAPH 2000, pages 403-410, 2000

[3] H. T. Bruce and P. Calder. Animating direct manipulation interfaces. In Proceedings of UIST '95, pages 3-12, 1995

[4] R. MacCracken and K. Joy. Free-form deformations with lattices of arbitrary

topology. In Proceedings of ACM SIGGRAPH 96, pages 181-188, 1996

[5] T. Milliron, R. Jensen, R. Barzel, and A. Finkelstein. A framework for geometric warps and deformations. ACM Transactions on Graphics, 21(1):20-51, 2002

[6] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. In Proceedings of ACM SIGGRAPH 86, 20(4):151-160, 1986

[7] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In Proceedings of ACM SIGGRAPH 2000, pages 165-172, 2000

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

[9] S. Forstmann, J. Ohya, A. Krohn-Grimberghe, and R. McDougall. Deformation styles for spline-based skeletal animation. In Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation, pages 141-150, 2007

[10] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. In Proceedings of ACM SIGGRAPH 91, pages 257-266, 1991

[11] S. F. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical report TR-97-19, Mitsubishi Electric Research Laboratories, 1997

[12] O. K. C. Au, C. L. Tai, L. Liu, and H. Fu. Mesh editing with curvature flow laplacian operator. Technical report, Computer Science Technical Report, HKUST-CS05-10, 2005

[13] J. Huang, X. Shi, X. Liu, K. Zhou, L. Wei, S. Teng, H. Bao, B. Guo, and H. Y. Shum. Subspace gradient domain mesh deformation. In Proceedings of ACM SIGGRAPH 2006, pages 1126-1134, 2006

[14] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In Proceedings of 3DPVT, pages 68-75, 2004

[15] M. Botsch, M. Pauly, M. Wicke, M. Gross. Adaptive space deformation based on rigid cells. In Proceedings of Eurographics 2007, 26(3):339-347, 2007

[16] D. Smythe. A two-pass mesh warping algorithm for object transformation and image interpolation. Tech. Rep. 1030, ILM Computer Graphics Department, Lucasfilm, San Rafael, Calif, 1990

[17] T. Ju, J. Warren, G. Eichele, C. Thaller, W. Chiu, and J. Carson. A geometric database for gene expression data. In SGP ’03: Proceedings of the 2003 Eurographics/ ACM SIGGRAPH Symposium on Geometry Processing, pages 166-176, 2003

[18] S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. In Proceedings of ACM SIGGRAPH 2006, 25(3):533-540, 2006

[19] M. Müller, B. Heidelberger, M. Teschner, M. Gross. Meshless deformations based on shape matching. In Proceedings of ACM SIGGRAPH 2005, 24(3):471-478, 2005

[20] A. R. Rivers and D. L. James. FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. In Proceedings of ACM SIGGRAPH 2007, 26(3):82, 2007

[21] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. In Proceedings of ACM SIGGRAPH 2005, 24(3):1134-1141

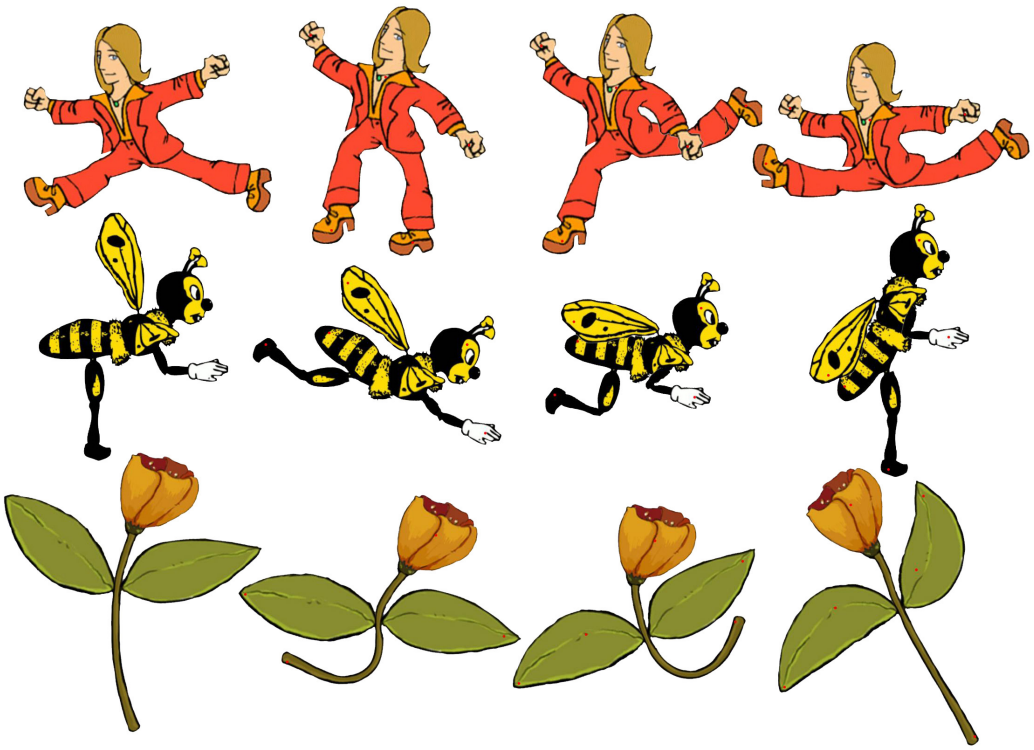


Figure 10: More results of our shape deformation method. In each row, the original shape is shown one the leftmost image and others are the deformation results.

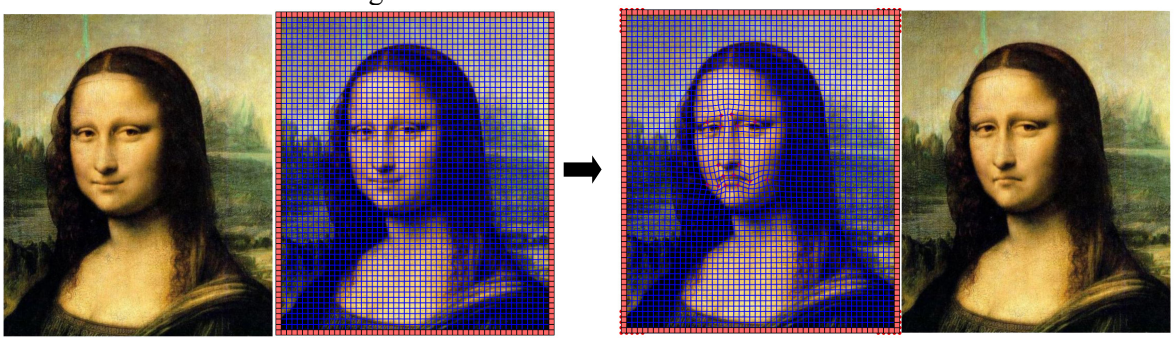


Figure 11: Image deformation for Mona Lisa by using our method. After deformation, her face is thinner and she is in a sad mood.

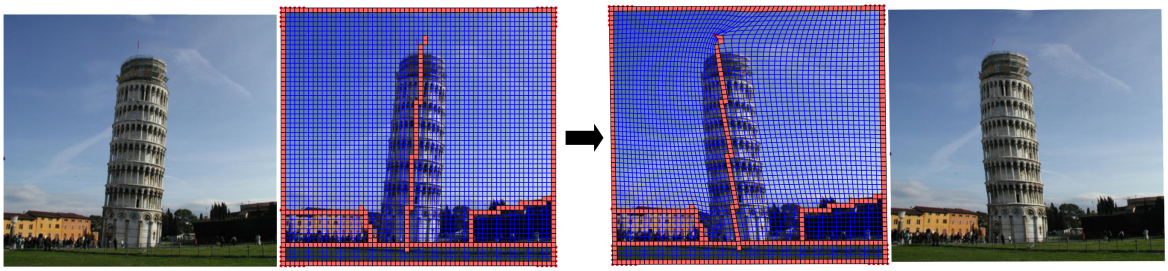


Figure 12: Image deformation for the Leaning Tower of Pisa by using our method. We implement deformation with line segment handles [18] by using the skeletal deformation method proposed in Section 4.2.