Vis Comput (2013) 29:545–553
DOI 10.1007/s00371-013-0817-1

ORIGINAL ARTICLE

Shape-aware skeletal deformation for 2D characters

Xun Wang · Wenwu Yang · Haoyu Peng · Guozheng Wang

Published online: 1 May 2013

Abstract This paper presents a skeleton-based method for deforming 2D characters. While previous skeleton-based methods drive the shape deformation by binding the skeleton to the shape, our method does so by propagating the skeleton transformations over the shape. In this way, the tedious process of weight selection in previous skeleton-based methods is not required. Also, the propagation allows us to consider the geometric characteristics of the shape such that local shape distortion can be effectively avoided. Experimental results demonstrate that our method allows real-time deformation and generates visually pleasing results.

Keywords Shape deformation · Skeleton · 2D Character Animation · Shape matching

1 Introduction

2D shape deformation is widely used in two-dimensional animation and image editing to represent moving objects of changing shape. Nowadays, this tool can be commonly found in commercial software for video editing or 2D character animation [4], such as Adobe After Effects, Adobe Flash, or Toon Boom Studio.

Recently, many deformation algorithms that allow the user to directly manipulate a shape through a click-and-drag interface have been introduced [8, 28]. By minimizing local distortion of the shape, these methods can generate pleasing and physically plausible deformation results. However, the click-and-drag interface may fail to control the natural deformation of 2D characters that have rigid limbs such as humans or animals (Fig. 1(b)).

For the characters that have a jointed structure, the skeleton-based approach, which uses a skeleton to control shape deformation, offers intuitive control as it naturally manifests the way in which the characters deform. In a typical workflow, the user defines a skeleton and the system binds the components of the character to the skeleton such that each component will follow motions of its associated skeleton bones: Each point on the character is transformed by a weighted linear combination of affine transformations defined by the associated bones [10]. However, binding the character to a skeleton is not a trivial task, and a tedious process of weight selection is typically required to obtain satisfactory deformation results [25].

In [25], Yan et al. decompose the shape into simplices, i.e., triangles (in 2D) or tetrahedra (in 3D), and use the skeleton to control the movement of simplices. In the method, each simplex is associated with only one skeleton bone, thus the weight selection issue is avoided. In [9, 22], the influence weights of each bone to the shape points are computed as discrete harmonic functions over a mesh coving the shape. Since such weights can spread the influences of the skeleton bones in a topology-aware and localized manner, the explicit binding between the shape and its skeleton is not needed. Essentially, however, the above methods determine the shape deformation mainly from the linear skeleton subspace, oblivious to the underlying geometric characteristics of the shape, which would possibly lead to inappropriate distortion (Figs. 2 and 7).

This paper presents a shape-aware 2D skeletal deformation method that combines the skeleton-based approach and the deformation algorithm that considers the shape rigidity. To make the shape deform naturally relative to the skeleton,

X. Wang · W. Yang (✉) · H. Peng · G. Wang
School of Computer Science & Information Engineering, Zhejiang Gongshang University, Hang Zhou, China
e-mail: wwyang@zjgsu.edu.cn

a propagation scheme is employed to smoothly spread the transformations of the skeleton bones over the shape. The propagation takes into account the geometric characteristics of the shape such that local shape distortion can be effectively avoided (Fig. 2(d)). Additionally, the propagation is automatic and the processes of binding and weight selection in previous skeleton-based methods are not required, thus the approach is not only easy to implement, but also easy to use for the user.

2 Related work

One popular approach for shape deformation is free form deformation (FFD) [7]. In FFD, a space deformation is defined via a (usually simple) control object. The user manipulates the control object and then the deformation of this object is interpolated to the entire 2D space. Generally, the lattices [17] and the so-called cages [13] are used as control objects. The FFD allows precise and flexible control; however, it typically requires aligning the control points of the control object with features of the shape, which can be cumbersome for the user.

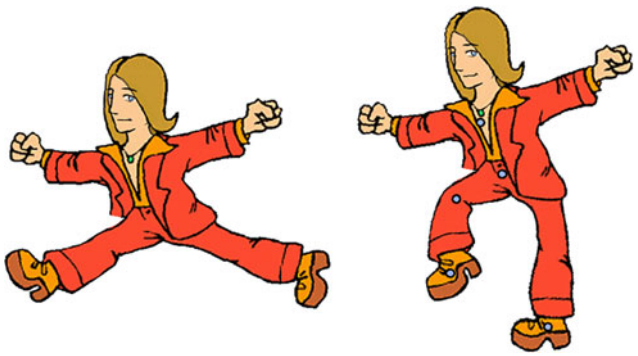

Fig. 1 Deformation of a 2D character: (**a**) input shape; (**b**) an unnatural pose generated by the method of [8], where a click-and-drag interface is used

Recently, a variety of 2D shape deformation algorithms [8, 21, 23, 26–28] that model the rigidity of the shape have been introduced (they can be considered as variants of detail-preserving differential mesh deformation techniques [3, 19, 29]). These approaches allow the user to directly manipulate a shape through a simple click-and-drag interface and generate physically plausible results by minimizing local shape distortion. However, as mentioned in the Introduction, the click-and-drag interface used in the methods does not take into account the natural way in which many objects' features are controlled, and thus may fail to generate natural deformation for 2D characters that have a jointed structure such as humanoid or robots. For these characters, their shapes and movement can be naturally described in terms of the motion of a skeleton. So an intuitive approach for manipulating such shapes is to use a predefined skeleton.

The traditional skeleton-based algorithms bind the shape to the skeleton and transform each point of the shape by a weighted linear blend of transformations associated with the skeleton bones influencing the point [12]. However, setting and tweaking these blending weights is extremely tedious if done manually. Yan et al. [25] decompose the shape into simplices and use the skeleton to drive simplex transformations, which avoids the use of weights. In [9, 22], the blending weights are automatically computed as discrete harmonic functions over a mesh coving the shape; they can produce smooth and intuitive deformation [11, 24, 30]. However, these methods have a common disadvantage, in that they do not take into account the underlying geometric characteristics of the shape, i.e., the rigidity of the shape, such that inappropriate distortion may occur.

In our work, the shape deformation is driven by the skeleton through a propagation scheme, which spreads the transformations associated with the skeleton bones over the shape in a smooth and shape-aware manner. The automatic prop-

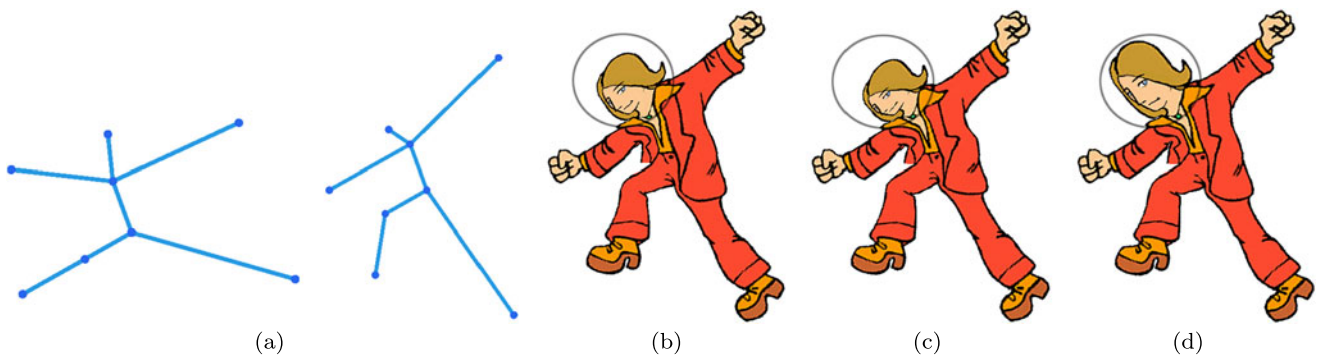

Fig. 2 2D character deformation using a skeleton: (**a**) initial and deformed skeletons for the character in Fig. 1(a), (**b**, **c**) results by the methods of [22] and [25], respectively, where the shape deformation is mainly derived from the linear skeleton subspace such that the character's head is distorted, and (**d**) result by our method where the distortion is effectively avoided

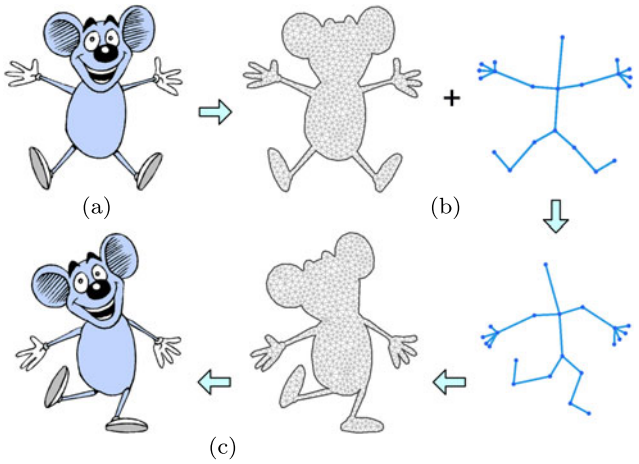

Fig. 3 Overview of the algorithm: (**a**) input shape, (**b**) triangulation and predefined skeleton, and (**c**) the user manipulates the skeleton to drive the triangle mesh deformation, which is then transferred to the shape

agation avoids the tedious processes of binding and weight setting and takes into account the geometric characteristics of the shape. Consequently, not only the intuitive and easily understood control using a skeleton is retained, but also pleasing and physically plausible results where local distortion is avoided are generated.

3 Overview

The input to our algorithm is a 2D shape, with the boundary represented as a simple closed polygon. A triangulated mesh is generated inside the boundary via constrained Delaunay triangulation [18] (Fig. 3b). The user then creates a skeleton (Fig. 3b), which consists of two or more bones meet at articulating joints, for controlling the shape's deformation.

The user manipulates the skeleton and the system deforms the shape accordingly (Fig. 3(c)). In our current implementation, the user can manipulate the skeleton in two modes: free mode and forward kinematics (FK) mode (see the accompanying video). In free mode, the user directly drags skeleton joints to arbitrary positions for desired skeleton pose. In FK mode, the user rotates the joints and the skeleton position is automatically located from the joint angles using forward kinematics scheme (see [5] for a description), which offers a natural way for controlling the skeleton's motion.

To deform the shape relative to the manipulated skeleton, a propagation scheme is employed to smoothly spreads the transformations associated with the skeleton bones over the whole triangle mesh. Then the deformation of the mesh is transferred to the embedded 2D shape by simple linear mapping (Fig. 3(c)). In the next section, we will describe the core algorithm, i.e., how to drive the triangle mesh deformation using the skeleton.

4 Algorithm

Similar to [25], we use the skeleton to drive the transformations of simplices, i.e., triangles of the mesh. Such technique includes two basic steps: (i) derive the simplex transformations from the transformations of skeleton bones and (ii) stitch together all of the transformed triangles into a whole using the original connectivity of the mesh. While [25] binds each mesh triangle to a skeleton bone such that the transformation of a triangle is directly determined by its associated bone, our method propagates the transformations of skeleton bones onto the mesh triangles in a progressive way. This has two main advantages:

- The explicit binding between the mesh triangles and the skeleton, which is still a challenging work, is not needed.
- The propagation can well incorporate the geometric characteristics of the shape, such that local distortion of the shape can be effectively avoided.

Notation Throughout, we use the following notation. We denote 2-vectors by boldface letters, while we denote 2×2 matrices by upper case letters. Furthermore, we denote a mesh vertex or a skeleton joint using an index, e.g., i or j. Specifically,

- $\overline{\mathbf{v}}_i$ and $\mathbf{v}_i$ denote the initial and deformed positions of each mesh vertex i, respectively.
- $\overline{\mathbf{p}}_j$ and $\mathbf{p}_j$ denote the initial and deformed positions of each skeleton joint j, respectively.

In the following, we describe at first the two basic steps mentioned above and then elaborate on the propagation scheme employed in our method.

4.1 Step one: deriving transformations of triangles

In the method, the transformations of mesh triangles are derived from the transformations associated with the skeleton bones. Thus, first the transformation matrix for each bone is calculated from the initial and deformed positions of the skeleton. Let $P = (\overline{\mathbf{p}}_1^b, \overline{\mathbf{p}}_2^b)$ and $Q = (\mathbf{p}_1^b, \mathbf{p}_2^b)$ be the initial and deformed positions of the two joints of a skeleton bone b, respectively, an affine transformation represented by matrix A that transforms P into Q can be calculated as follows.

In order to calculate A, an additional, virtual joint, which locates on the line that is perpendicular to the bone, is associated to the bone b (Fig. 4). The initial position of the joint is: $\overline{\mathbf{p}}_3^b = (\overline{\mathbf{p}}_1^b + \overline{\mathbf{p}}_2^b)/2 + \overline{\mathbf{p}}_1^b\overline{\mathbf{p}}_2^{b\perp}$, where $\overline{\mathbf{p}}_1^b\overline{\mathbf{p}}_2^b = \overline{\mathbf{p}}_2^b - \overline{\mathbf{p}}_1^b$ and $\perp$ is a 2D operator such that: $(x, y)^\perp = (-y, x)$. We transform the virtual joint using the scale-free version of the bone's transformation, thus the deformed position of the joint can be determined: $\mathbf{p}_3^b = (\mathbf{p}_1^b + \mathbf{p}_2^b)/2 + s \times \mathbf{p}_1^b\mathbf{p}_2^{b\perp}$, where s is the scale factor in the direction perpendicular to the bone. While s is set as 1 at default in our implementation, the user

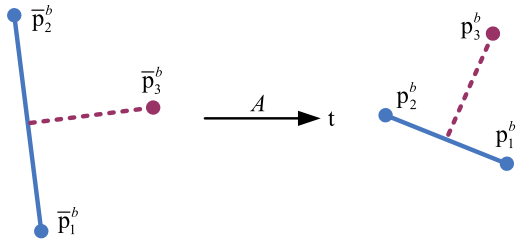

Fig. 4 Computation of a bone's transformation by adding a virtual joint. Note that the scaling in the direction perpendicular to the bone is 1 at default

can modify it for the desired scaling effect. Then the transformation matrix of the bone b can be obtained from the following expression:

$$A\overline{\mathbf{p}}_j^b + \mathbf{t} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \overline{\mathbf{p}}_j^b + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{p}_j^b, \quad j \in \{1, 2, 3\}$$

Note that we do not take the translation $\mathbf{t}$ into account since it does not change any property of mesh triangles excepting for their positions, which can also simply be determined from the skeleton joints.

We now derive the transformations of mesh triangles from the bones' transformations. Denote the mesh triangles as $\mathcal{T} = \{T_{\{i,j,k\}}\}$ and their transformations $\{A_{\{i,j,k\}}\}$. To determine $\{A_{\{i,j,k\}}\}$, we classify the triangles into two categories. One includes the triangles that are intersected with the bones, and the other includes those that are not intersected with the bones. For each triangle in the first category, its transformation is set the same as the transformation of its intersecting bone: If two or more bones intersect with the triangle, a random bone is selected. Then the bones' transformations applied to the triangles in the first category are propagated over the mesh, such that the transformations of the triangles in the second category are also obtained. The propagation process, which takes into account the geometric characteristics of the shape, is described in Sect. 4.3 in detail.

4.2 Step two: stitching transformed triangles

For each mesh triangle $T_{\{i,j,k\}}$, we transform it from its initial position $\{\overline{\mathbf{v}}_i, \overline{\mathbf{v}}_j, \overline{\mathbf{v}}_k\}$ to a new position $\{\mathbf{v}_i^{\text{new}}, \mathbf{v}_j^{\text{new}}, \mathbf{v}_k^{\text{new}}\}$ by its associated transformation, i.e., $A_{\{i,j,k\}}$. Since most of the mesh vertices lie in more than one triangle, a vertex may (in general) be transformed to several different positions by the transformations associated with each triangle in which it appears. To determine the final deformed positions of the mesh vertices, we stitch together all of the transformed triangles by minimizing the following quadratic error function, which is similar to the assembly process in [1, 8, 20]:

$$E_s = \sum_{\{i,j,k\} \in \mathcal{T}} w_{\{i,j,k\}} \sum_{\substack{(l,m) \in \{(i,j),\\ (j,k),(k,i)\}}} \left\| \mathbf{v}_l \mathbf{v}_m - \mathbf{v}_l^{\text{new}} \mathbf{v}_m^{\text{new}} \right\|^2$$

where $w_{\{i,j,k\}}$ is a penalty factor for the deviation between the actual transformed position of each triangle and its ideal transformation, and could be used to control the local stiffness of the shape. Note that because the transformations $\{A_{\{i,j,k\}}\}$ do not carry translations of the triangles, we define an error on each edge vector, not the position of each vertex.

In order to determine the actual position of the deformed mesh, we further take the translation information into account by incorporating the skeleton joints' positions. For each joint j, we represent it using a linear combination of the vertices of the mesh triangle in which the joint lies, i.e., $\mathbf{p}_j = C_j V_j$, where C_j is a row vector that is composed of the barycentric coordinates of the joint relative to the triangle in their initial configurations, and V_j is a column vector composed of three 2D vectors corresponding to the deformed vertex positions of the triangle. Then we define a positional constraint for the deformed mesh, which preserves the positional consistency between the shape and the skeleton:

$$E_p = \sum_j \| C_j V_j - \mathbf{p}_j \|^2$$

Finally, we minimize the following quadratic function to obtain the deformation results, where the unknown variables are the vertex coordinates of the deformed mesh:

$$\arg\min_{\mathbf{v}_i} \quad E_s + w_p E_p \tag{1}$$

We use the weight $w_p = 100$ for all the examples shown in the paper. Equation (1) corresponds to a classical quadratic optimization problem. By computing the gradients of Eq. (1) with respect to each $\mathbf{v}_i$ and setting them to zero, we transform Eq. (1) into a sparse linear system:

$$MV = b \tag{2}$$

where V is a vector composed of all unknown vertex positions $\mathbf{v}_i$. Since the coefficient matrix M is only dependent on the initial configurations of the mesh and the skeleton, the full factorization can be reused and during shape deformation only two back substitutions are required for solving Eq. (2), resulting in a very efficient solver [6].

4.2.1 Tuning of stiffness weights

In the term E_s of Eq. (1), the weight $w_{\{i,j,k\}}$ for each triangle is provided to control the local stiffness of the shape. In default, all of the weights are set to 1. As discussed in Sect. 4.1, for the triangles in the first category, i.e., those that are intersected with the skeleton bones, their transformations are set the same as the transformations of their intersecting bones, such that the shape parts corresponding to these triangles will tightly follow the skeleton during deformation, which is necessary for intuitive manipulation. However, the uniform weights in E_s may not ensure the actual transformations of

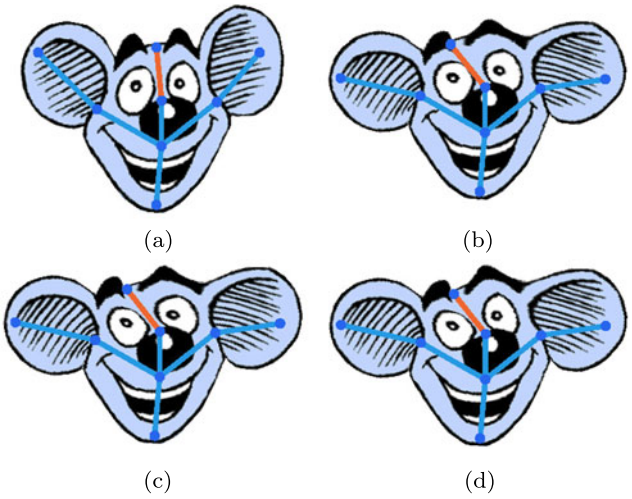

Fig. 5 Deformation with stiffness weights: (**a**) input shape and skeleton; (**b**) deformation with uniform weights where the shape movement is inconsistent with the bone that is being manipulated, i.e., the one in orange; (**c**) the default weights of the triangles intersecting with the bones are increased; (**d**) the eye of the rat on the right side is preserved by increasing the weights of corresponding triangles to 50

these triangles are coincident with the bones' transformations (Fig. 5(b)). Thus, we increase the default weights to 100 for the triangles in the first category such that the transformations imposed on these triangles can be well retained in minimizing Eq. (1) (Fig. 5(c)). Additionally, we provide an intuitive painting interface in which the user can make certain shape parts stiffer or softer than others by adjusting the weights. This is useful in preserving the shape of specified features (Fig. 5(d)).

4.3 Shape-aware propagation of skeleton's motion

As shown in Sect. 4.1, the mesh triangles are classified into two categories and the bones' transformations are directly applied to the triangles in the first category, i.e., those that are intersected with the skeleton bones. We now discuss how the bones' transformations are propagated to the rest triangles, i.e., those in the second category, while taking into account the geometric characteristics of the shape.

4.3.1 Motivation

Denote the mesh triangles in the first and second categories as $\{T^1_{\{i,j,k\}}\}$ and $\{T^2_{\{i,j,k\}}\}$, respectively, and their transformations $\{A^1_{\{i,j,k\}}\}$ and $\{A^2_{\{i,j,k\}}\}$, respectively. While $\{A^1_{\{i,j,k\}}\}$ are directly obtained from the bones' transformations, we employ a propagation scheme to spread the influences of the bones to the triangles in $\{T^2_{\{i,j,k\}}\}$, such that $\{A^2_{\{i,j,k\}}\}$ are determined. The propagation is motivated by the observation that the transformations of neighboring triangles should be consistent with respect to one another; otherwise distortion would occur in the triangles of the stitched mesh (Fig. 6(b)). The basic idea is to progressively update $\{A^2_{\{i,j,k\}}\}$ by minimizing distortion of the triangles $\{T^2_{\{i,j,k\}}\}$ in the stitched mesh, such that all the triangle transformations are consistent with the bones' transformations.

4.3.2 Updating of transformations

Initially, all of $\{A^2_{\{i,j,k\}}\}$ are set as an identity matrix. With the current configurations of $\{A^1_{\{i,j,k\}}\}$ and $\{A^2_{\{i,j,k\}}\}$, the mesh triangles are transformed and then stitched together as discussed in Sect. 4.2. Obviously, the transformations of the triangles intersecting with the bones that have been manipulated are quite different from the identity matrix, thus are inconsistent with the transformations of their neighboring triangles in $\{T^2_{\{i,j,k\}}\}$, leading to triangle distortion in the current deformed mesh (Fig. 6(b)). To avoid such distortion, we update $\{A^2_{\{i,j,k\}}\}$ to make the transformations of all the mesh triangles as consistent as possible. To update $\{A^2_{\{i,j,k\}}\}$, we use the shape matching technique [14], which can effectively avoid the distortion of their associated triangles. Given a transformation $A^2_{\{i,j,k\}}$ and its associated triangle $T^2_{\{i,j,k\}}$, the shape matching problem can be stated as follows: Let $\{\overline{\mathbf{v}}_i, \overline{\mathbf{v}}_j, \overline{\mathbf{v}}_k\}$ and $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ be the vertex positions of the triangle $T^2_{\{i,j,k\}}$ in the initial and current deformed meshes, respectively. Find the rotation matrix R, which minimizes

$$\sum_{l=\{i,j,k\}} \left\| \mathbf{v}_l - \mathbf{v}^c - R\left(\overline{\mathbf{v}}_l - \overline{\mathbf{v}}^c\right) \right\|^2$$

where $\overline{\mathbf{v}}^c$ and $\mathbf{v}^c$ are the rotation centers of the triangle in the initial and current deformed configurations, i.e., $\overline{\mathbf{v}}^c = (\overline{\mathbf{v}}_i + \overline{\mathbf{v}}_j + \overline{\mathbf{v}}_k)/3$ and $\mathbf{v}^c = (\mathbf{v}_i + \mathbf{v}_j + \mathbf{v}_k)/3$. In 2D case, the optimal rotation matrix R has an analytic solution [15]. We then update the transformation of each triangle in $\{T^2_{\{i,j,k\}}\}$, i.e., $A^2_{\{i,j,k\}}$, to the corresponding optimal rotation matrix.

Using the similar process mentioned above, transformations of $\{A^2_{\{i,j,k\}}\}$ are updated several times such that all of them become more and more consistent with the bones' transformations (Figs. 6(b)–6(d)). Consequently, a final deformed mesh with least distortion is obtained (Fig. 6(d)). It bears noting that the stitching optimization of Eq. (1) will spread the inconsistency between transformations of the triangles over the whole mesh [25], which can also reduce triangle distortion in the stitched mesh to some extent. Hence, a small number of updating is sufficient in general. In our experiments, the number of times of updating is about 12.

5 Results

Table 1 lists the performance statistics of the examples in the paper. In the table, "vertex" denotes the vertex number

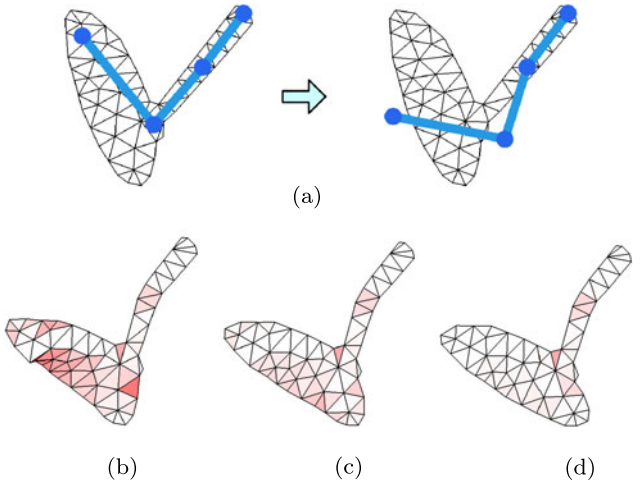

Fig. 6 Distortion reduction by updating the transformations of mesh triangles. (**a**) Initial mesh and skeleton (*left*). To drive the mesh deformation, the skeleton is manipulated to a new position (*right*), (**b**–**d**) stitched triangle meshes during the updating of triangle transformations, where $\{A^2_{\{i,j,k\}}\}$ are initially set as an identity matrix in (**b**), and are then updated 6 times in (**c**), and are finally updated 12 times in (**d**). The triangles are rendered according to their distortion. The triangle distortion is described by area change, which is calculated by $|\triangle A|/A$ where A is the original triangle area and $\triangle A$ is the triangle area change after deformation. Note that the lighter the triangle color, the less the triangle distortion

Table 1 Performance statistics (PC with 3 GHz Intel Core 2 Duo CPU and 2 GB RAM, single threaded). Timing is measured in milliseconds

Figure	Vertex (num)	PreComput (ms)	Solve (ms)
2(d)	526	5.0	5.9
3	672	5.9	6.6
7(d)	522	4.7	5.0
8	1053	10.6	11.2
9	273	2.2	2.5
10	869	9.4	10.9

of the underlying mesh of the shape, while "pre" and "solve" denote the runtime of pre-computation and the solving time of deformation, respectively. The runtime cost of mapping the deformation of the mesh to the shape is not included as it only corresponds to several milliseconds for all examples, and thus could be negligible. Obviously, the shape deformation can be performed interactively.

Figures 1 and 2(d) demonstrate that our skeleton-based deformation method is more suitable for the characters that have a jointed structure, such as humans or animals, than the direct manipulation methods using a click-and-drag interface [2, 8]. For these characters, our method, which uses a skeleton to control shape deformation, offers intuitive control as it naturally manifest the way in which the characters deform.

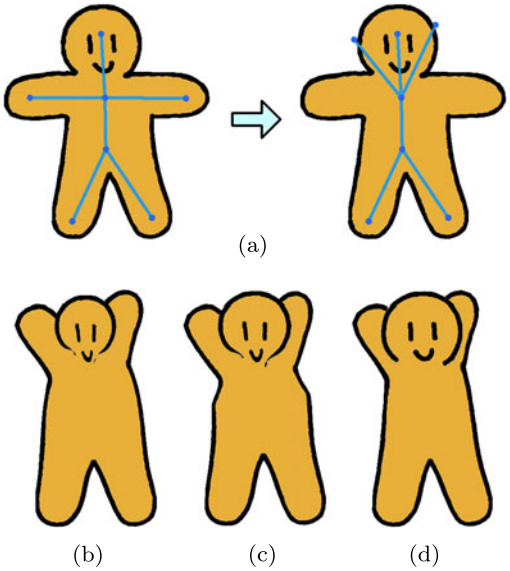

Fig. 7 2D character deformation using a skeleton. (**a**) input shape and skeleton, (**b**, **c**) results by the methods of [22] and [25], respectively, and (**d**) result by our method

Figures 2 and 7 demonstrate that our method generates more visually pleasing results than previous skeleton-based approaches [22, 25]. The previous approaches derive the shape deformation mainly from the linear skeleton subspace, which may lead to inappropriate distortion. In Fig. 7(a), the three skeleton bones, corresponding to the head and hands of the baby, split the baby's head into two local linear subspaces. Obviously, lifting the bones of the baby's hands squashes the two subspaces. Consequently, as shown in Figs. 7(b)–7(c), the baby's head located in the two subspaces is distorted. However, our method takes into account the geometric characteristics of the shape such that local shape distortion can be effectively avoided (Fig. 7(d)).

In Figs. 8, 9 and 10, we have applied our method to more examples. Figure 8 further demonstrates that our method offers intuitive and natural control and generates visually pleasing results. Figure 9 demonstrates that our method enables the user to locally edit a shape with only setting several bones on the parts that are expected to be edited. It bears noting that our method is also suitable for the shape that has no obvious jointed structure (Fig. 10).

Animations Using the proposed method, we can make 2D animations by setting the skeleton pose of the character at each key frame: The skeletons are then interpolated between keyframes to generate a smooth sequence of deforming skeletons, which are used to drive the shape deformation. To interpolate the skeletons, a straightforward approach is to directly interpolate the positions of the skeleton joints; however, this tends to produce shrinkages, especially when large rotations occur. Thus, similar to [16], we avoid shrinkages by interpolating the intrinsic parameters of the skeletons.

Fig. 8 2D character deformation using a skeleton: (**a**) input shape and its skeleton, (**b**, **c**) results generated by our method

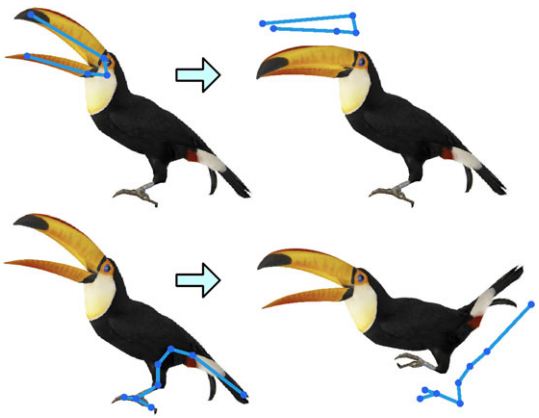

Fig. 9 Local skeletal shape deformation. (*Left*) input shape and skeleton; (*right*) deformed skeleton and shape

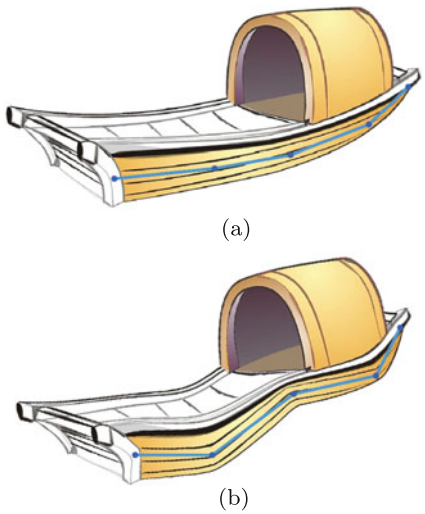

Fig. 10 Deformation of a boat: (**a**) input shape and skeleton, (**b**) deformed skeleton and shape

For each skeleton, its intrinsic parameters include: joint angles and bone lengths, and can be calculated using the forward kinematics theory [5]. We then interpolate the corresponding intrinsic parameters as well as the positions of the skeleton's root joint between keyframes using the parametric method of curve fitting. At each interpolating time, the intermediate skeleton can be computed from its intrinsic parameters and root joint position using the forward kinematic scheme [5]. Finally, we obtain a smooth sequence of deforming skeletons, which ultimately leads to a natural transition of character shapes between keyframes (see Fig. 11 and the accompanying video).

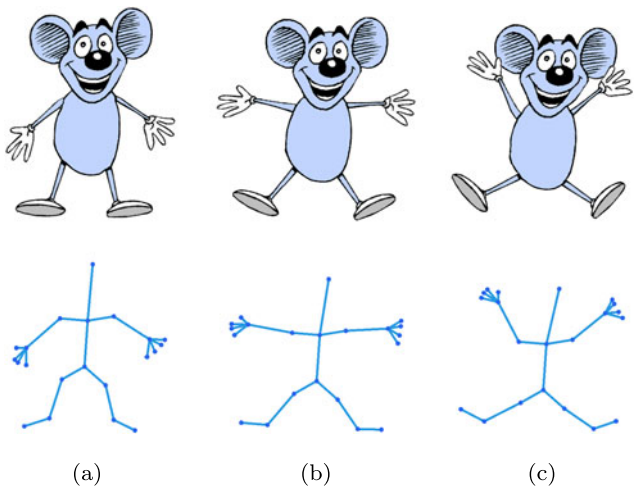

Fig. 11 Interpolation of character shapes between two keyframes: (**a**, **c**) shapes and skeletons in the keyframes, (**b**) interpolated skeleton and shape at time $t = 0.5$

6 Conclusion

We have presented a 2D skeletal deformation method that takes into account the geometric characteristics of the shape. The method allows the user to control shape deformation using a skeleton. To drive the shape deformation using the skeleton, we introduce a propagation scheme, which smoothly spreads the transformations of the skeleton bones

over the shape. The propagation allows us to consider the geometric characteristics of the shape, i.e., the rigidity of the shape, such that local shape distortion can be effectively avoided. Our method avoids the tedious processes of binding and weight setting in previous skeleton-based methods and generates visually pleasing results.

Considering the future work, we hope to apply the method to 3D characters. The main problem seems to be the increasing computational cost of the optimizations required in our method. The computational cost of the optimization is dependent on the vertex number of the mesh inside the shape's boundary. While the vertex number of 2D triangle mesh is typically small or moderate, the vertex number of 3D tetrahedron mesh is generally large due to an additional dimension in 3D space, and thus cannot guarantee the real time manipulation. We intend to investigate the possibility of replacing the 3D tetrahedron mesh with the adaptive 3D lattice.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments. This research was partially funded by the Natural Science Foundation of China (Nos. 61003189, 61170098), the National Basic Research Program of China (No. 2009CB320801), the Natural Science Foundation of Zhejiang Province (Nos. LY12F02025, Z1101243), the Science and Technology Agency projects of Zhejiang Province (Nos. 2012C33074, 2012R10041-16), and the National High Technology Research and Development Program of China (863 Program, No. 2013AA013701).

References

1. Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: SIGGRAPH '00, pp. 157–164 (2000)
2. Botsch, M., Pauly, M., Wicke, M., Gross, M.: Adaptive space deformations based on rigid cells. Comput. Graph. Forum **26**(3), 339–347 (2007)
3. Botsch, M., Sorkine, O.: On linear variational surface deformation methods. IEEE Trans. Vis. Comput. Graph. **14**(1), 213–230 (2008)
4. Bregler, C., Loeb, L., Chuang, E., Deshpande, H.: Turning to the masters: motion capturing cartoons. ACM Trans. Graph. **21**(3), 399–407 (2002)
5. Craig, J.: Introduction to Robotics: Mechanics and Control. Addison–Wesley Series in Electrical and Computer Engineering: Control Engineering. Prentice Hall, New York (2005)
6. Davis, T.A.: Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. ACM Trans. Math. Softw. **30**(2), 196–199 (2004)
7. Gain, J., Bechmann, D.: A survey of spatial deformation from a user-centered perspective. ACM Trans. Graph. **27**(4), 1–21 (2008)
8. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Graph. **24**(3), 1134–1141 (2005)
9. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. **30**(4), 78:1–78:8 (2011)
10. Jacobson, A., Sorkine, O.: Stretchable and twistable bones for skeletal shape deformation. ACM Trans. Graph. **30**(6), 165 (2011)
11. Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T.: Harmonic coordinates for character articulation. ACM Trans. Graph. **26**(3), 71 (2007)
12. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: SIGGRAPH '00, pp. 165–172 (2000)
13. Lipman, Y., Levin, D., Cohen-Or, D.: Green coordinates. ACM Trans. Graph. **27**(3), 1–10 (2008)
14. Müller, M., Heidelberger, B., Teschner, M., Gross, M.: Meshless deformations based on shape matching. ACM Trans. Graph. **24**(3), 471–478 (2005)
15. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Trans. Graph. **25**(3), 533–540 (2006)
16. Sederberg, T.W., Gao, P., Wang, G., Mu, H.: 2-D shape blending: an intrinsic solution to the vertex path problem. In: SIGGRAPH '93, pp. 15–18 (1993)
17. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. Comput. Graph. **20**(4), 151–160 (1986)
18. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and delaunay triangulator. In: Applied Computational Geometry: Towards Geometric Engineering. Lecture Notes in Computer Science, vol. 1148, pp. 203–222. Springer, Berlin (1996)
19. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: SGP'04, pp. 175–184 (2004)
20. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004)
21. Sýkora, D., Dingliana, J., Collins, S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In: NPAR '09, 25–33 (2009)
22. Weber, O., Sorkine, O., Lipman, Y., Gotsman, C.: Context-aware skeletal shape deformation. Comput. Graph. Forum **26**(3), 265–274 (2007)
23. Weng, Y., Xu, W., Wu, Y., Zhou, K., Guo, B.: 2d shape deformation using nonlinear least squares optimization. Vis. Comput. **22**(9), 653–660 (2006)
24. Xu, K., Zhang, H., Cohen-Or, D., Xiong, Y.: Dynamic harmonic fields for surface processing. Comput. Graph. **33**(3), 391–398 (2009)
25. Yan, H.B., Hu, S., Martin, R.R., Yang, Y.L.: Shape deformation using a skeleton to drive simplex transformations. IEEE Trans. Vis. Comput. Graph. **14**, 693–706 (2008)
26. Yang, W., Feng, J.: 2d shape manipulation via topology-aware rigid grid. Comput. Animat. Virtual Worlds **20**(2–3), 175–184 (2009)
27. Yang, W., Feng, J., Jin, X.: Shape deformation with tunable stiffness. Vis. Comput. **24**(7–9), 495–503 (2008)
28. Yang, W., Feng, J., Wang, X.: Structure preserving manipulation and interpolation for multi-element 2D shapes. Comput. Graph. Forum **31**(7/2), 2249–2258 (2012)
29. 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. **23**(3), 644–651 (2004)
30. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Comput. Graph. Forum **24**(3), 601–609 (2005)

Xun Wang is currently a professor at the School of Computer Science and Information Engineering, Zhejiang Gongshang University, People's Republic of China. He received his Ph.D. degree from the State Key Lab of CAD&CG, Zhejiang University in 2006. His research interests include mobile graphics computing, image/video processing, computer animation, geographic information systems, and intelligent information processing.

Wenwu Yang is currently an associate professor at the School of Computer Science and Information Engineering, Zhejiang Gongshang University, People's Republic of China. He received his Ph.D. degree from the State Key Lab of CAD&CG, Zhejiang University in 2009. His research interests include cartoon animation, image editing, computer graphics, and geometric modeling.

Haoyu Peng is currently an associate professor at the School of Computer Science and Information Engineering, Zhejiang Gongshang University, People's Republic of China. He received his Ph.D. degree from the State Key Lab of CAD&CG, Zhejiang University in 2007. His research interests include computer graphics and computer vision.

Guozheng Wang is currently an associate professor at the School of Computer Science and Information Engineering, Zhejiang Gongshang University, People's Republic of China. He received his Bachelor's degree from Zhejiang Gongshang University in 1985. His research interests include computer graphics and image processing.