Sketch-Based Skeleton-Driven 2D Animation and Motion Capture

Junjun Pan and Jian J Zhang

National Centre for Computer Animation, Media School,
Bournemouth University, UK
{pjunjun,jzhang}@bournemouth.ac.uk

Abstract. We present a novel sketch-based 2D animation technique, which allows the user to produce 2D character animations efficiently. It consists of two parts, sketch-based skeleton-driven 2D animation production and 2D motion capture. The user inputs one image of the character and sketches the skeleton for each subsequent frame. The system deforms the character and creates animations automatically. To perform 2D shape deformation, a variable-length needle model is introduced to divide the deformation into two stages: skeleton driven deformation and nonlinear deformation in joint areas. It preserves the local geometric features and global area. Compared with existing approaches, it reduces the computation complexity and produces plausible results. Because our technique is skeleton-driven, the motion of character can be captured by tracking joints position and retargeted to a new character. This facilitates the reuse of motion characteristics contained in existing moving images, making the cartoon generation easy for artists and novices alike.

Keywords: sketch, skeleton, cartoon, 2D shape deformation, motion capture.

1 Introduction

Sketch-based animation has gained increasing popularity in the field of computer graphics due to its intuitiveness and importance as a useful tool for character modeling and animation. Many papers [1,2,3] have been published and several techniques have been developed into commercial software, e.g. [4]. With the help of sketch-based techniques, animators can translate their 2D drawings directly into 3D models. Instead of handling the detail step by step, the modeler/animator can visualize and evaluate the fast-prototyped models at an early stage, which can be further refined with other 3D tools to meet the practical needs. However, compared with the progress in 3D animation, 2D animation has not benefited as much from these advantages. Most professional cartoon studios still produce huge amounts of animation (key-frames and in-betweens) manually [5], which is a laborious and time-consuming process. The generation of key-frames and in-between frames are the two most important and labor intensive steps in 2D animation production. To best use the animators time, the key-frames are drawn by skillful *key-framers*, while the in-betweens by those who are less experienced and skillful, known as the *in-betweeners*. Although some software tools, e.g. Animo, Toon Boom [6], have been helpful in generating in-between frames, they often lack of 'personality' in comparison with those created by a human in-betweener. The software-generated in-betweens have to be tweaked by the animator to give back the 'personality' to the animation. In practice, many in-betweens remain created manually.

Motivated by the skeleton-driven 3D animation techniques and some recent progress in 2D deformations, e.g. [7], in this paper we present a new technique aiming to improving the degree of automation for the production of 2D animation without sacrificing the quality. Our method consists of two parts, Part 1: 2D animation sequence generation and Part 2: motion capture and retargeting. Part 1 can be used independently to create an animation sequence. If it is combined with Part 2, one can easily reuse the 'motion' of an existing animation sequence and apply it to a different character. The primary application of our technique is 2D animation production. But it is also applicable to interactive graphical systems where the user can deform a 2D shape directly by moving its skeleton. Since it is very simple to use, we anticipate that this method is not only of interest to professional cartoon production houses, but also to novices for creating 2D moving graphics.

The most important issue concerning Part 1 is to handle the complex shape deformation of characters both realistically and efficiently. For a character at a given orientation (for example, side view, front view or back view), we first generate its skeleton by analyzing the geometry of the boundary curve. Similar to a 3D character, the skeleton acts as the driving structure and controls the deformation of the character. To deform a character, we introduce the so called *variable-length needle model* and propose an algorithm called *skeleton driven + nonlinear least squares optimization*. The idea is to divide the 2D shape deformation into two components. The first is skeleton driven deformation, which is controlled purely by the corresponding segment of the character skeleton; and the other is nonlinear least squares optimization, which is to compute the deformation in the joint areas which are associated with the skeletal joints. Our observation suggests during animation most complex deformation occurs around the joint areas of a character. For the interest of computational efficiency, the skeleton driven deformation is treated simply as a linear transformation. Only the deformation in the joint areas is solved by nonlinear least squares optimization. To ensure realistic deformation, properties such as boundary features and local area preservation are maximized during animation. The property of global area preservation is also easily achieved by the *variable-length needle model*. Therefore once the first frame is given, the animator can easily create an animation sequence by drawing the skeleton for each subsequent key-frame. The system will produce the deformed character shape automatically, saving the animator from drawing the whole frame.

Although large amounts of video, cartoon and traditional 2D moving images exist, few effective approaches are available to make use of these abundant resources due to the special characteristics and principles of 2D animation [8,9]. The main objective of Part 2 is to patch this obvious gap. Because our cartoon production technique is skeleton-based, we can naturally borrow the idea of motion capture from 3D animation to capture the 'motion' of a 2D animation sequence. In 3D animation, the skeleton length of a 3D character is usually constant during animation. However, in a 2D case, changing feature lengths in the form of squash and stretch is one of the most powerful and expressive principles of animation [8]. In this paper we will demonstrate that with our method we can use the 2D skeleton to represent these important and expressive transformations.

Retargeting the captured motion to a different character has been extensively studied in 3D animation, e.g. [10]. We present a feature region based tracking method, commonly used in computer vision, to extract the motion of 2D objects in video or an image sequence. We apply a mixed optimization strategy coupled with template matching and Kalman prediction. Once the user has located all the joint regions of a character in the first frame, the system will track the motion of the joints automatically in the subsequent frames. The captured motion information is then retargeted to the predefined skeleton of a new 2D character to generate the deformation (animation). What to be noted is tracking is well studied in computer vision and our purpose here is not to develop a new tracking method. The novelty is to use this technique to capture 2D motion, which up to now remains an unsolved issue. To our knowledge, no effective 2D motion capture methods exist, which are good enough for 2D animation production.

There are three key contributions in this paper:

1. We present a sketch-based skeleton-driven 2D animation technique for cartoon characters. To produce a new key-frame, the user only needs to sketch the skeleton.
2. To handle 2D shape deformation, we have developed a *variable-length needle model* and introduced the *skeleton driven + nonlinear least squares optimization* algorithm. Compared with other approaches, it is more efficient and able to produce plausible deformation with squash-and-stretch effects.
3. We introduce a straightforward skeleton-based 2D motion capture method which can extract the motion from cartoon, video and rendered moving image sequences by tracking the motion of joints. Using both geometric and visual features, it prevents self-occlusion and feature disappearance in moving images.

The remainder of this paper is organized as follows: the related work is discussed in Section 2. Our sketch-based skeleton-driven 2D animation technique is described in Section 3, while in Section 4 we describe the motion capture method. Section 5 gives the experimental results and comparison with previous approaches. The limitations and possible improvements in future will be discussed in Section 6.

2 Related Work

There is a significant body of previous work concerning 2D character animation [7,11,12,13]. Here we only discuss the most relevant developments including 2D shape deformation and motion capture.

2D shape deformation: Most recent 2D deformation techniques are *control point* based. Although skeletons are incorporated into some commercial packages, the purpose is primarily to help pose a character, not to deform or animate a character [6]. Igarashi et al. [7] designed an “as-rigid-as-possible” animation system which allows the user to deform the shape of a 2D character by manipulating some control points.

To reduce the cost, the authors presented a two step deformation algorithm, which simplifies it into two linear least-squares minimization problems. As it only approximates the original problem, it can produce implausible results due to its linear feature. Weng et al. [13] presented a 2D shape deformation algorithm based on nonlinear least squares optimization. The authors used a non-quadratic energy function to represent this problem, which achieves more plausible deformation results. However, the iterative solution is computationally more costly. Schaefer et al. [14] proposed a 2D shape deformation algorithm based on linear moving least squares. It avoids input image triangulation and performs smooth deformation globally. They also extended this point-based deformation method to line segments. However, as the authors admitted, this method deforms the entire image with no regard to the topology of the object. This weakness limits its use in 2D character animation. Wang et al. [15] presented another 2D deformation technique based on the idea of rigid square matching. Instead of using triangular meshes, they use uniform quadrangular meshes as the control meshes. As the obtained deformation is quite rigid, it is not a perfect fit for soft objects and the global area is not preserved.

All above methods employ global optimization. One disadvantage of such global optimization is that the shape of all triangles needs re-computing even if a small pose change happens. This is computationally expensive and is not necessary in many cases. In our implementation, we divide the shape deformation into two components: skeleton driven deformation and nonlinear deformation of the joint areas. The former can be treated as a linear transformation and the latter is solved by nonlinear least squares optimization, but only for local regions. This local optimization scheme reduces the computation costs and can still achieve plausible deformation results.

Motion capture and retargeting: Most research on motion capture and retargeting focuses on 3D animation [10,16]. Many effective algorithms have been developed and benefited numerous applications including computer games and film special effects. In contrast, little has been done for 2D animation. Bregler et al. [17] presented a method to capture and retarget the non-rigid shape changes of a cartoon character using a combination of affine transformation and key-shape interpolation. It is effective in representing the qualitative characteristics (i.e. motion in this case). But it is difficult to be precise. Therefore, although it can be useful for cartoon retargeting, it is not easy for the animator to control the movement and deformation accurately. In contrast, a skeleton-driven approach gives the animator better control of the deformation during animation. Hornung et al. [18] presented a method to animate photos of 2D characters using 3D motion capture data. Given a single image of a character, they retarget the motion of a 3D skeleton to the character's 2D shape in image space. To generate realistic movement, they use "as-rigid-as-possible" deformation [7] and take projective shape distortion into account. In comparison, our method directly transfers the 2D motion data from an existing image sequence. We don't require 3D motion data. Also it does not need the user to manually specify the correspondence between 2D and 3D poses of a character. Sykora et al. [19] proposed an image registration method by combining locally optimal block matching with as-rigid-as-possible shape regularization. It can be used to motion capture a 2D object. However, the limitation is it cannot handle occlusion or large deformation.

2D animation can be regarded as a consistent image sequence. Our approach, which is influenced by several video based approaches [20,21,22], tracks the motion of the character's joints. However, since our system needs dealing with a variety of characters with different shape and topology, the model-based tracking methods are ineffective. We choose more general features: texture (colour) and geometry information (position, velocity) of the joints to extract the motion of a character. Comparing with the KLT tracker [20], not relying on good feature selection, our algorithm directly tracks the interested feature regions (joints) for each frame.

3 Sketch-Based Skeleton-Driven 2D Animation

Our technique consists of five steps. We use a popular cartoon figure, mm (Fig. 1a), to illustrate the technique.

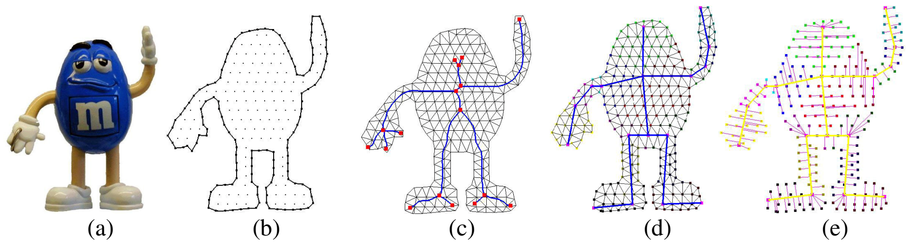

Fig. 1. Initial preprocessing before deformation. (a) Original template model, (b) Silhouette detection and discrete sampling, (c) Triangle mesh and curve skeleton, (d) Skeleton and decomposition, (e) The variable-length needle model.

3.1 Silhouette Detection and Triangulation

The user first imports a 2D character serving as the *original template model*, which can be represented by a BMP/JPEG image or vector graphics. The requirement is that the boundary of the object should be represented by a closed polygon. For BMP/JPEG images, we currently remove the background manually. Its silhouette is detected with the marching squares algorithm [23], forming a closed polygon. Distributing discrete points allows the polygon to be triangulated. Many triangulation algorithms exist. Here we adopt the Constrained Delaunay triangulation algorithm. The sampling density is adjustable at the user's will to form sparser or denser meshes depending on the requirements. To make sure a character shape is properly triangulated, we require the template model should be expanded or the limb occlusion is solved beforehand. This can be performed with image completion [24].

3.2 Skeletonization and Decomposition

The process of constructing a skeleton is called the *skeletonization*. The system first generates a *curve skeleton* of the character with the 2D thinning algorithm [25]. To produce an *animation skeleton*, the user locates the joints either on the curve skeleton or the mesh vertices. The curve skeleton of the example character is shown in Fig. 3c.

Some end points of the curve skeleton branches (red points in Fig. 3c) can be used as skeletal joints directly. After skeletonization, the system attaches every vertex to its nearest skeleton segment. This is called the *decomposition*, which classifies the vertices into different *regions*. Here we use a region growing algorithm described in [26]. The decomposition result for the example cartoon character is shown in Fig. 3d. In this figure, there are 16 skeleton segments, which have been colour-coded to represent the associated *vertex regions*.

Based on the classification of all the vertices, we now classify the triangles into two types, *interior triangles* and *joint triangles*. If the three vertices of a triangle are of the same color, i.e. they are all associated with one skeleton segment, the triangle is an interior triangle, otherwise the triangle is a joint triangle. Both types of triangles are shown in Fig. 1. We also sort the vertices into three categories, *silhouette vertices, interior vertices* and *joint vertices* illustrated in Fig. 2. Silhouette vertices form the contour of an object. Except for silhouette vertices, if all the neighbor triangles of a vertex are interior triangles, this vertex is an *interior vertex*; otherwise it is a *joint vertex*.

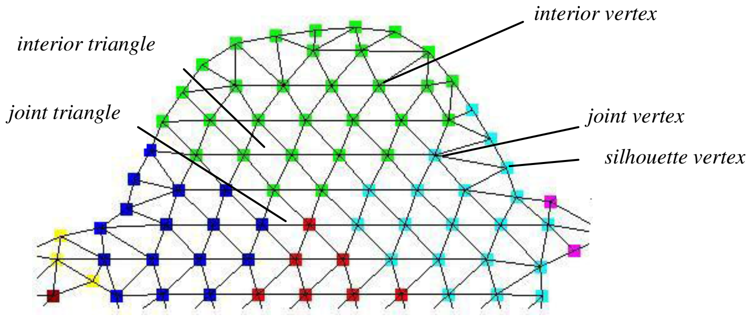

Fig. 2. Illustration of our definition of different types of vertices and triangles

3.3 Shape Deformation

Shape deformation is crucial to the quality of animation and is an essential step of our technique. The main objective for our algorithm design is both to minimize the boundary change, interior shape distortion and computational overheads. We deform a 2D character in two stages: skeleton driven deformation for each vertex region (Stage 1) and nonlinear deformation for the joint areas (Stage 2). For Stage 1, since the computation involves simple transformations, it incurs only a small overhead. Stage 2 minimizes implausible deformations. Although the computation is more complex, it involves only a small portion of the vertices.

3.3.1 Variable-Length Needle Model

Our Variable-Length needle model represents the geometry of the deformable object using a collection of variable-length needles. Each needle links a vertex to its attached skeleton segment. Each needle originates from the skeleton and extends outward in a fixed angle. The vertex is at the end point of a needle. The length of a needle is the Euclidean distance between the vertex and the corresponding skeleton segment. Fig. 1e illustrates the variable-length needles model.

3.3.2 Stage One: Skeleton Driven Deformation

In skeleton driven deformation, the geometry of all vertices is determined only by the position of the corresponding skeleton segment. Because the points are close to the skeletal segment, it is reasonable to regard the needles as being subject to the affine transformations of the skeleton segment during animation. Rotation and scaling are legitimate transformations here. During transformation, the length and direction of the needles relative to the skeleton segment are unchanged, leading to fast computation of the new coordinates of the mesh vertices.

Cartoon characters often exhibit significant squash-and-stretch deformations. An advantage of using our needle model is that the area enclosed by the boundary can be maintained by ensuring the change of the length of a needle to be reciprocal of the change of the linked skeletal segment length. Because the needles cover the character's surface, this simple method effectively preserves the global area of the character and express the squash-and-stretch effects. Fig. 3 demonstrates the effect of global area preservation. One skeletal segment is used to deform the bottle.

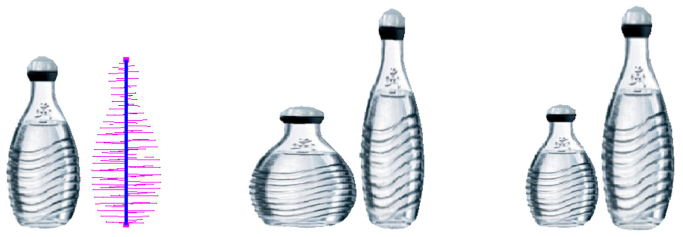

Fig. 3. Deformation with (middle) and without (right) global area preservation. The original object and variable-length needle model are shown on the left.

Fig. 4 illustrates the deformation process of a cartoon character. As can be seen in Fig. 4c, d, the deformation is realistic. However, the texture and contour curve in some joint areas are not sufficiently smooth, and some joint triangles even overlap. This suggests that to minimize shape distortion, we need to concentrate on the joint areas and ensure the deformation conforms to the original model. This forms the main part of Stage two.

3.3.3 Stage Two: Nonlinear Deformation in Joint Areas

We employ two geometric entities as constraints to prevent shape distortion: rotation and scale invariant (RSI) Laplacian coordinates [27] and edge lengths of the triangular mesh. The former preserves the local shape feature of the contour curve and the latter for local area preservation.

Let (V, E) be the 2D graph of a character's mesh model, where V and E are the sets of vertices and edges respectively. V can be divided into three subsets: V_s which contains k silhouette vertices, V_p which contains m joint vertices, and V_q which contains $n-m-k$ interior vertices.

a. RSI Laplacian coordinates

As the ordinary Laplacian coordinates do not account for rotation and scaling of the curve, here we use rotation and scale invariant (RSI) Laplacian coordinates [27] to handle the deformation of the silhouettes. Given that we are mainly interested in the joint areas where visible distortions occur, we only need to constrain the silhouette vertices in the joint areas, denoted by $V_{s'}$. To preserve the local features of the contour curve, we need to minimize the following objective function:

$$\sum_{v_i \in V_{s'}} \| T(v_i) - T(\tilde{v}_i) \|^2 \tag{1}$$

where $T(v_i)$ stands for the RSI Laplacian coordinates of v_i before deformation; $T(\tilde{v}_i)$ stands for the RSI Laplacian coordinates of v_i after deformation.

b. Edge lengths

We use the following energy function to penalize edge length deviation for joint triangles:

$$\sum_{v_i, v_j \in V_p, (i,j) \in E} \| |v_i - v_j| - |\tilde{v}_i - \tilde{v}_j| \|^2 \tag{2}$$

$|v_i - v_j|$ is the edge length of $v_i v_j$ before deformation, and $|\tilde{v}_i - \tilde{v}_j|$ is the edge length of $v_i v_j$ after deformation.

Combining (2) and (3), our overall objective function can be rewritten in the following matrix form:

$$w_1 \| \mathbf{T}\mathbf{V}_{s'} - \mathbf{T}\tilde{\mathbf{V}}_{s'} \|^2 + w_2 \| \mathbf{H}\mathbf{V}_\mathbf{p} - \mathbf{H}\tilde{\mathbf{V}}_\mathbf{p} \|^2 \tag{3}$$

Assume the number of vertices in $V_{s'}$ is k'. $\mathbf{V}_{s'}$ represents the coordinates of these vertices. $\mathbf{H}$ is a $|E_p| \times m$ matrix, which is used to compute the edge vectors of joint triangles. The sum of weights: w_1 and w_2 are normalized to 1 and in our experiments we used equal weightings for both terms. However, the user can adjust the weighting to emphasize certain geometric properties.

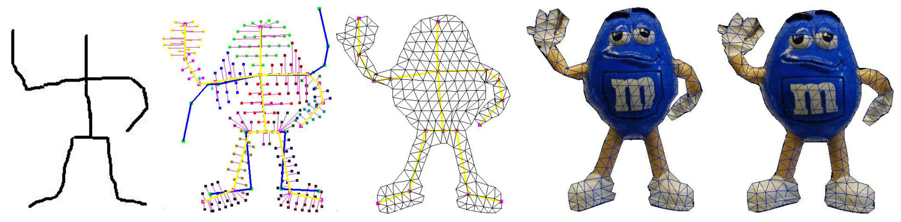

Fig. 4. Deformation process. (a) Sketched skeleton, (b) Deformed character displayed as a variable-length needle model. The blue lines represent the skeleton of the original template model before deformation, (c) Mesh and skeleton after the deformation of Stage one, (d) Character after the deformation of Stage one, (e) Character after the deformation of Stage two.

This is a non-linear function and to solve the optimization problem efficiently, we adopt the iterative Gauss-Newton method. The result is shown in Fig. 4e where both the silhouette and texture inside the object are smoothly deformed compared with the result of Stage one. For this particular example, the computation converges with 36 iterations. The number of iterations varies with many factors including the shape of model, the number of vertices and the magnitude of the deformation. In our experiment, the average number of iterations across all the examples is 35.

3.4 Depth Adjustment and Fine Tuning

Collision detection is a practical problem for the deformation of cartoon characters. When different parts of a character overlap, if the depths are not assigned properly, the overlapping parts may interpenetrate. Moreover, assigning static depth values for vertices [7] does not work in all possible situations. In our system, we allow dynamic depth adjustment through interaction. Upon the generation of a new deformed model, we monitor the mesh for self-intersection and set an appropriate depth order to the overlapping parts. When the user clicks any vertex in an overlapping part, all the vertices in this decomposed region will have the same depth value as the clicked one. Fig. 5a gives an example of depth adjustment.

Our system also allows the user to fine tune the local geometric details of the model in two ways: sketch curves and point dragging. The sketch curves are used to fine tune the silhouette of an object. Similar to the nearest neighbor method, we search the start and end points of the silhouette segment along the object contour (the shortest Euclidean distances from the start and end points respectively to the sketch curve). For each vertex on the silhouette segment of the variable-length needle model, we fix the angle between the needle and the skeleton segment, and change the length of the needle to move its end point to the new position on the sketch curve. An example is given in Fig. 5b where the profile of the right arm is altered with a sketch curve. Point dragging is more straightforward. The user picks and drags any vertex to reshape the character. It can be very useful to edit or generate detailed shape changes after the skeleton-driven deformation is complete, such as facial expressions. Fig. 5c shows two examples. The left one changes the face expression and the right one creates a hedgehog hair.

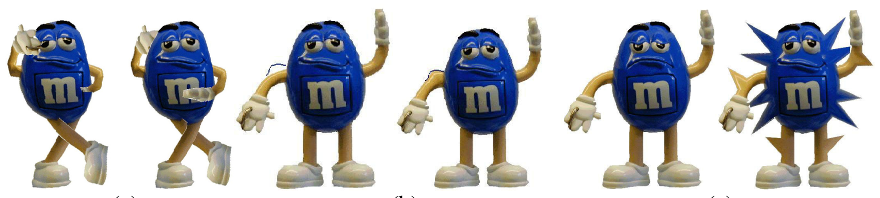

Fig. 5. Depth adjustment and fine tuning local geometric detail. (a) Deformed result before and after depth adjustment, (b) Sketch curve fine tuning, (c) Deformation through point dragging.

3.5 In-Betweening

In-between frames are generated by interpolating the deformation produced from the two stages discussed above, skeleton-driven deformation (stage 1) and non-linear deformation in the joint areas (stage 2). Many interpolation techniques can be used. In this Section, we explain how to generate the in-betweens given two key-frames. Suppose f_{start}, $f_{skeleton\text{-}driven}$ and f_{end} represent the shape of the initial frame before deformation, the shape generated with the skeleton-driven deformation only and the shape of the end frame, respectively. The computation of each in-between frame *f(t)* consists of two elements. The first describes the skeleton-driven deformation which is solved by spherical linear interpolation (slerp). The second element represents the non-linear deformation which can be computed by the linear interpolation of the geometry displacement between $f_{skeleton\text{-}driven}$ and f_{end} . The formula can be described as follows:

$$\underset{t\in[0,1]}{f(t)} = \text{slerp}[f_{start}\times(1-t)+f_{skeleton-driven}\times t]+(f_{end}-f_{skeleton-driven})\times t \tag{4}$$

4 Motion Capture and Retargeting

Based on the method proposed above, we have also developed an effective algorithm to capture the motion of a 2D character. The basic idea is to track the joints using the well developed computer vision techniques. Once the first frame is identified from a moving image sequence, the curve skeleton is automatically extracted in the same way as was described earlier. Based on this curve skeleton, the animator marks the joints on the image. To capture the motion from the subsequent frames/images, the key step is to track the positions of the joints. Because we are concerned with 2D images/frames, it is reasonable to assume the texture of the joints unchanged between any two adjacent frames. Our design therefore is to track the joint positions using texture as the visual cue. It captures the motion of an *original character* and retargets it to the *target character*. To ensure it works correctly, the image sequences and the target character should satisfy the following preconditions:

1. The image sequence is consistent, i.e. the change between two adjacent frames is relatively small.
2. The target character has the same topology and a similar pose to that of the original character in the first frame.

3. The pose of the original character in the first frame is roughly expanded. There is no occlusion for all the joints.

What needs pointing out is that our motion capture method is not limited to cartoon sequences only. It can capture a cartoon sequence, a video and a rendered 3D animation image sequence (Fig. 12).

4.1 Tracking

For a given image sequence or video as input, the system first subtracts the background for each frame [28]. The user then locates all the joints by marking small rectangles on the original character to indicate the joint positions, using the automatically generated curve skeleton as a guide. Fig. 6a shows an *original character* to be tracked. The red rectangles represent the located joint regions. Tracking and connecting all the joint positions in these frames lead to the generation of the skeleton in the subsequent frames. To map the captured motion to a target character (Fig. 6b), we require the target character to have a similar topology and pose to those of the original character. Moving images of static objects can be relatively easy to track with color information alone. But it is not sufficient for articulated characters. This is because parts of a character may overlap from time to time where color information disappears. In order to solve this problem, in addition to the color feature as discussed, we also use the geometric feature. The geometric feature allows the joint positions to be predicted in the next frame by estimating the velocity of the joints.

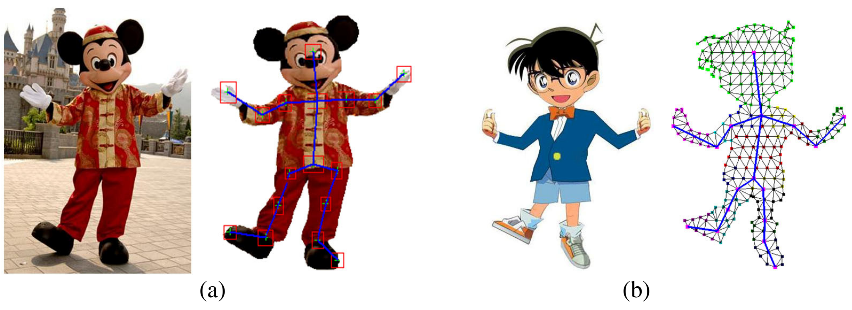

Fig. 6. Initial setup for motion capture. (a) Original character in the first frame and located joints, (b) Target character and its decomposition results.

Assume n joints to be tracked in each frame, the positions of the rectangle centres at frame t form a geometric feature vector $\mathbf{G}_t = [\mathbf{g}_{1t}, \mathbf{g}_{2t}, \ldots, \mathbf{g}_{mt}, \ldots \mathbf{g}_{nt}]^T$ $=[(x_1, y_1)_t, (x_2, y_2)_t, \ldots, (x_m, y_m)_t, \ldots (x_n, y_n)_t]^T$. For the visual feature, we use an n dimensional feature vector $\mathbf{C}_t = [\mathbf{c}_{1t}, \mathbf{c}_{2t}, \ldots, \mathbf{c}_{mt}, \ldots \mathbf{c}_{nt}]^T$, where $\mathbf{c}_{mt}$ is the texture matrix of the m-th rectangle region. We track a joint (the centre of the corresponding rectangle) between adjacent frames by searching the closest match in the previous frame. Using the Bayes' rule with a uniform *a priori* distribution case, this process is equivalent to finding the maximum of $P(\mathbf{F}_t \mid \mathbf{\Theta})$, where $\mathbf{F}_t$ denotes a feature vector of the character at frame *t*. $\mathbf{\Theta}$ denotes the feature parameters corresponding to the tracked result at frame *t*-1. Here the whole feature space is divided into two sub-spaces: geometric and visual spaces as follows:

$$P(\mathbf{F}_t \mid \mathbf{\Theta}) = P_c(\mathbf{C}_t \mid \mathbf{\Theta}_c) P_g(\mathbf{G}_t \mid \mathbf{\Theta}_g) \tag{5}$$

where $P_c(\mathbf{C}_t \mid \mathbf{\Theta}_c)$ and $P_g(\mathbf{G}_t \mid \mathbf{\Theta}_g)$ are PDFs (probability density functions) corresponding to the visual and geometric features respectively. Maximizing $P(\mathbf{F}_t \mid \mathbf{\Theta})$ can be described as the following optimization problem, which is to minimize the sum of the Mahalanobis distances in the sub-spaces, i.e.

$$\min \sum_{m=1}^{n} D_{mt} \tag{6}$$

s. t. $D_{mt} = w_\alpha D_{c,mt} + w_\beta D_{g,mt}$, $D_{g,mt} = (x_{mt} - \bar{x}_{mt})^2 + (y_{mt} - \bar{y}_{mt})^2$

$$D_{c,mt} = w_r RedDiff(\mathbf{c}_{mt}) + w_g GreenDiff(\mathbf{c}_{mt}) + w_b BlueDiff(\mathbf{c}_{mt})$$

where w_α and w_β are the weights used to normalize the corresponding distances. In our work, w_α is $1/(255)^2$ and w_β is $(1/r)^2$. r is the radius of the search range. $D_{c,mt}$ represents the measuring distance in RGB space. $D_{g,mt}$ represents the distance between the centre of moving rectangle and the position of the predicted joint region centre.

The Kalman filter is widely used for tracking as a subject of computer vision. Since the interval between adjacent frames is small in our work, we treat it as a uniformly accelerated motion in a time interval and use the following prediction model to compute the centres of the joint regions $\mathbf{G}_t(\bar{x}_m, \bar{y}_m)$

$$\begin{aligned} \mathbf{G}_t(\bar{x}_m, \bar{y}_m) &= \mathbf{G}_{t-1}(x_m, y_m) + \mathbf{V}_{m,t-1}T + \mathbf{A}_{m,t-1}T^2/2 \\ \mathbf{V}_{m,t-1} &= [\mathbf{G}_{t-1}(x_m, y_m) - \mathbf{G}_{t-2}(x_m, y_m)]/T \\ \mathbf{A}_{m,t-1} &= [\mathbf{V}_{t-1}(x_m, y_m) - \mathbf{V}_{t-2}(x_m, y_m)]/T \end{aligned} \tag{7}$$

where $\mathbf{G}_{t-1}(x_m, y_m)$, $\mathbf{V}_{m,t-1}, \mathbf{A}_{m,t-1}$ are the tracked centre position, velocity and acceleration of the *m*-th joint at frame *t*-1. *T* is the interval between adjacent frames.

Fig. 7a illustrates the joint tracking result of the character in Fig. 6a. We select four tracked frames in an 18 frame image sequence. As can be seen from frames 6, 12, 18, the problem of self-occlusion is effectively solved with our position prediction algorithm.

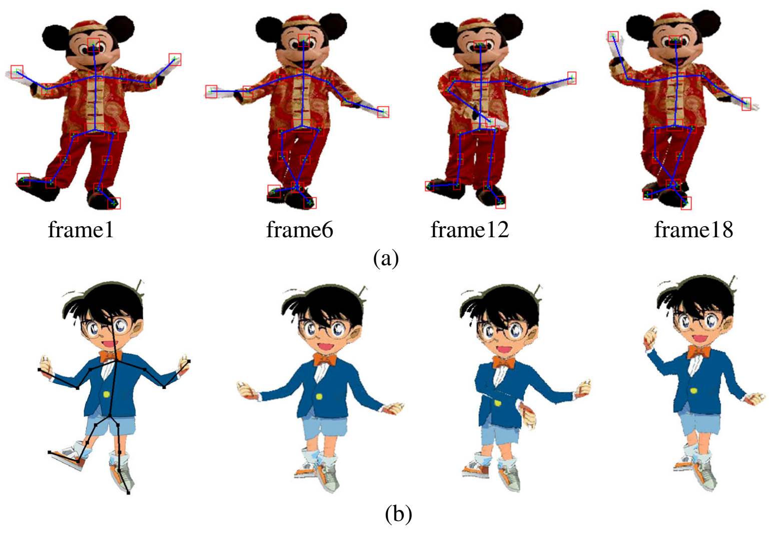

Fig. 7. Tracking and retargeting. (a) Joint tracking for the original character, (b) Deformed target character.

This tracking method is not without limitations. Since the frame-by-frame tracking is inherently subject to error accumulation, the accuracy is limited to a small number of frames (around 30 in our experiments). One effective way to solve this problem is to divide a large image sequence into a number of segments which consist of fewer frames, and correct the tracking error for the first frame in each segment. Our system allows the user interactively adjust the tracking result at any frame when necessary.

4.2 Retargeting

To retarget a captured motion to the new character, we first produce a skeleton as described before. There is a lot of existing work on 3D animation, such as [10], which is directly applicable to our case. In this paper however, we only implemented a simple method to demonstrate the retargeting process. For a moving 2D character, a skeleton can have both linear (length) and angular (orientation) displacements, i.e. a skeleton segment can stretch / squash and rotate. The basic idea of our simple motion retargeting is to map the captured increments of both length and orientation angle of a skeletal segment, which can be computed by:

$$\Delta l_{m,t} = l_{m,t} / l_{m,t-1}, \Delta\alpha_{m,t} = \alpha_{m,t} - \alpha_{m,t-1} \tag{7}$$

where $l_{m,t}$, $\alpha_{m,t}$ represent the length and orientation angle of the m-th skeleton segment at frame t. For the target model, the length $l'_{m,t}$ and orientation angle $\alpha'_{m,t}$ of the m-th skeleton segment at frame t can be trivially computed by:

$$l'_{m,t} = l'_{m,t-1}\Delta l_{m,t}, \alpha'_{m,t} = \alpha'_{m,t-1} + \Delta\alpha_{m,t} \tag{8}$$

Fig. 7b illustrates the retargeting result for the target character in Fig. 6b.

5 Experiments and Evaluation

We design two experiments to comparatively study the computational complexity and visual performance of our deformation algorithm. The first is deforming a 2D flower model with our algorithm into four similar postures to those in [13]. The results are shown in Fig. 8. We test our algorithm on a 3.2GHz Pentium 4 workstation with 1GB memory. Table 1 gives the comparison results. Since our deformation algorithm does not perform nonlinear shape deformation for all triangles, it takes about a quarter of the time. This is especially significant when performing larger and more complex animations.

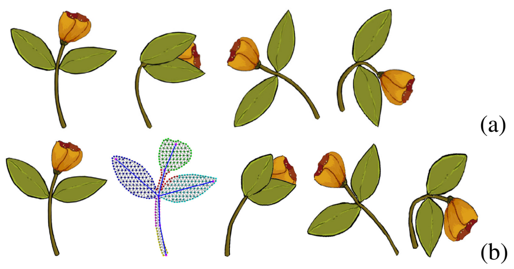

Fig. 8. Flower model deformed by our algorithm and [13]. (a) Deformation results with [13], (b) Deformation results with our algorithm (from left to right, original template, decomposition result and deformed figures).

Table 1. Comparison of data statistics and timing

Cartoon model: Flower	Method in [13]	Our deformation algorithm
Boundary vertices	114	123
Interior vertices	256	27 (Joint vertices)
Precomputing time	22ms	9ms
Iteration time	0.589ms	0.143ms

The second experiment is to deform an elastic object both appeared in [7] and [15]. Two skeletal segments are used in our algorithm. Fig. 9 gives the results. As our method preserves the global area, comparing with the result in [7] and [15], it can express the squash-and-stretch effect of this elastic object naturally during deformation.

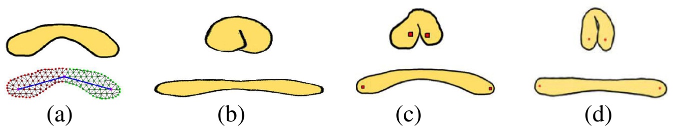

Fig. 9. Comparing our algorithm with the approaches in [7] and [15]. (a) Original object and decomposition result with skeleton, (b) Deformation result with our algorithm, (c) Deformation result in [7], (d) Deformation result in [15].

We also invited three animators to test our technique with two groups of experiments. The first was used to evaluate the visual quality and performance of animation production. The original characters were acquired from the Internet. Fig. 10 and the video give the results. The second group is to test our motion capture and retargeting method. There are two experiments. The first (Fig. 11) is to track the joints of a jumping cartoon man and retarget the motion to a new character. We treat the hat and the man as two objects, and track them separately. The second one (Fig. 12) is to track the joints of a 3D running horse (rendered as a 2D image sequence using Maya) and retarget it into a cartoon gazelle.

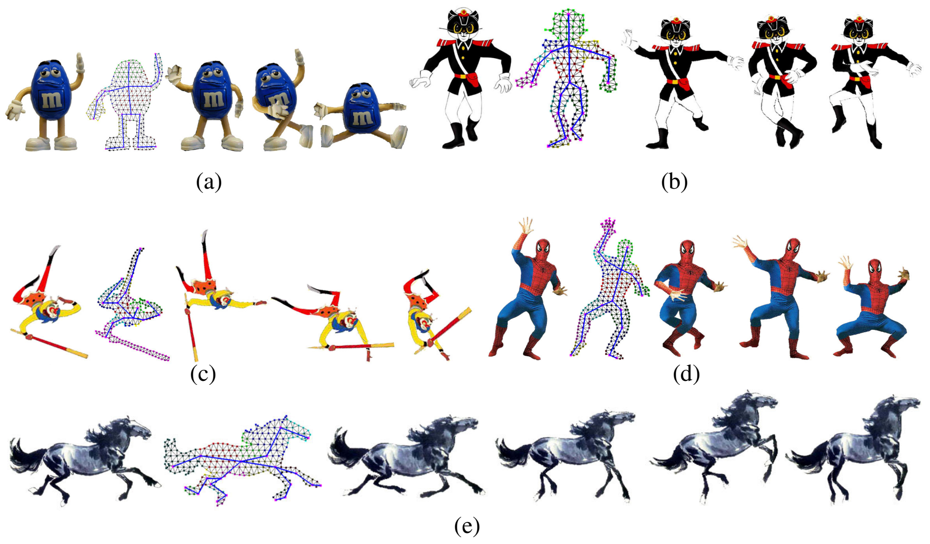

Fig. 10. Five groups of cartoon characters deformed by our algorithm. From left to right, original template model, decomposition results with skeleton, deformed figures. (a) mm, (b) Black cat sergeant, (c) Monkey king, (d) Spiderman, (e) Running horse.

The consensus from the animators showed that our method is more efficient than the current practice adopted in many commercial cartoon production houses, as sketching a skeleton is much faster than drawing a whole frame. In fact it is encouraging to see that our design is consistent with their animation practice. To create a key-frame, often the animator would first sketch a stick figure (i.e. the skeleton) and then overlay the body shape on top guided by the stick figure. This process is called the *deep structure*. Sketching the skeleton alone relieves them from some of the time-consuming tasks, i.e. to draw the whole character body. They also believe that our motion capture technique will make an animator's life much easier and have a positive impact on the cartoon production industry once a 2D motion database is established.

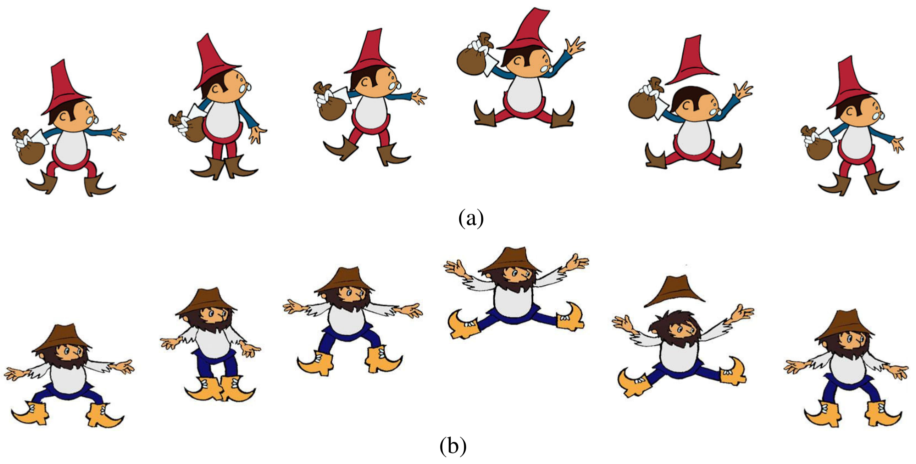

Fig. 11. Motion of a jumping cartoon man retargeted to a new character. (a) Original cartoon character, (b) Retargeted new character.

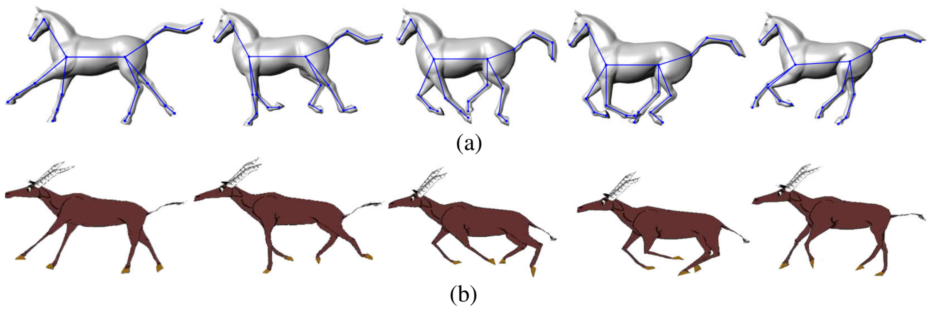

Fig. 12. Joint tracking of a running horse (a) and retargeting to a cartoon gazelle (b)

6 Discussion and Limitations

In this paper, we have presented a sketch-based skeleton-driven 2D animation technique using sketches as the primary inputting means both for the creation and the control of the animation artifacts. It consists of two main parts. The first is concerned with the fast production of 2D character animation by sketching only the skeletons. Comparing with the traditional cartoon production pipeline, drawing a skeleton is much faster than drawing a whole frame. This allows denser key-frames to be drawn by experienced animators. By reducing the interval between key-frames, in many cases the in-between frames can be produced mainly by software without compromising the realism, unlike the current practice where human in-betweeners are the main workforce, which is often expensive.

Given an original image of a character and the sketched skeleton sequence, our technique will generate a deformed character with different poses automatically. It is faster and less labour-intensive than the existing production practice. Our theoretical contribution in this aspect includes a variable-length needle model, which successfully preserves the global area of a character during animation, which is an essential property for squash-and-stretch effect in cartoon animation; and the skeleton driven + nonlinear least squares optimization algorithm, which is computationally economic.

The second part of our work is concerned with the development of a skeleton-based 2D motion capture technique. Once a skeleton is established in the first frame of a moving image sequence, we track all the joint positions from each subsequent image considering both geometric and visual features of the images. This 2D motion capture technique can be applied to various types of moving images, including 2D cartoon animation, videos and image sequences of rendered 3D animations.

Our research also reveals some limitations of the developed method. The first relates to the texture information of the template image. Because there is no 3D information of a 2D character, large pose change can result in loss of correct texture for subsequent frames. Although some research in matting, image completion and texture synthesis [18,24,29,30] has attempted to resolve this issue, it is still an open problem for all 2D deformation techniques. We plan to use image merging techniques to tackle it in the future. The second limitation is the error accumulation in tracking. Currently we correct the tracking error at the first frame of each sequence segment. We plan to use a more robust tracking approach in the future. The third place to improve is retargeting. Our current simple approach is only to demonstrate our motion capture method. It would be desirable to incorporate a 3D animation technique (e.g. [10]) to treat retargeting as a space-time optimization problem. The motion editing techniques developed for 3D motions are also relevant.

Acknowledgments. This research is in part supported by EPSRC grant EP/F030355/1.

References

1. Davis, J., Chuang, E., Slesin, D.: A sketching interface for articulated figure animation. In: Proc. Eurographics/ SIGGRAPH Symposium on Computer Animation, pp. 320–328 (2003)
2. Thorne, M., Burke, D., Panne, M.: Motion Doodles: An interface for sketching character motion. In: SIGGRAPH 2004 Conference Proceedings, pp. 424–431 (2004)
3. Li, Y.: 3D Character Animation Synthesis From 2D Sketches. In: Proceedings of the 4th international Conference on Computer graphics and Interactive Techniques in Australasia and Southeast Asia, pp. 81–90 (2006)
4. Igarashi, T., Matsuoar, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: Proceedings of ACM SIGGRAPH 1999, pp. 79–89 (1999)
5. Sykora, et al.: Sketching cartoons by example. In: Eurographics Workshop on Sketch-Based Interfaces and Modeling, pp. 27–34 (2005)
6. ToonBoom: `http://www.toonboom.com/products/digitalpro/eLearning/tipsTricks/2008`
7. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Graphics 24(3), 1134–1141 (2005)

8. Williams, R.: The Animator's Survival Kit. Faber & Faber, London (2001)
9. Isaac, K.: Applying the 12 Principles to 3D Computer Animation. The Art of 3D Computer Animation and Effects (2003)
10. Gleicher, M.: Retargeting motion to new characters. In: Proceedings of ACM SIGGRAPH 1998, pp. 33–42 (1998)
11. Hsu, S.C., Lee, I.H.: Drawing and animation using skeletal strokes. In: Proceedings of ACM SIGGRAPH 1994, pp. 109–118 (1994)
12. Fekete, J., Bizouarn, E., Cournarie, E.: TicTacToon: A paperless system for professional 2D animation. In: ACM SIGGRAPH 96 Conference Proceedings, pp. 79–90. (1996)
13. Weng, Y., Xu, W., Wu, Y., Zhou, K., Guo, B.: 2D Shape Deformation Using Nonlinear Least Squares Optimization. The Visual Computer 22(9-11), 653–660 (2006)
14. Schaefer, S., Mcphail, T., Warren, J.: Image deformation using moving least squares. In: SIGGRAPH 2006 Conference Proceedings, pp. 255–262. (2006)
15. Wang, Y., Xu, K., Xiong, Y., Cheng, Z.: 2D shape deformation based on rigid square matching. Computer Animation and Virtual Worlds 19(3-4), 411–420 (2008)
16. Fayreau, L., Reveret, L., Depraz, C., Cani, M.P.: Animal gaits from video. In: Eurographics/SIGGRAPH Symposium on Computer Animation, pp. 277–286 (2004)
17. Bregler, C., Loeb, L., Erika, Chuang, E. Deshpande, H.: Turning to the masters: motion capturing cartoons. In: SIGGRAPH Conference Proceedings, pp 121–129. (2002)
18. Hornung, A., Dekkers, E., Kobbelt, L.: Character animation from 2D pictures and 3D motion data. ACM Trans. Graph. 26(1), 1–9 (2007)
19. Sykora, et al.: As-Rigid-As-Possible Image Registration for Hand-drawn Cartoon Animations. In: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering, pp. 25–33 (2005)
20. Kong, Y., Rosenfield, A.: Digital topology: Introduction and survey. Comp. Vision, Graphics and Image Proc. 48(3), 357–393 (1989)
21. Drori, I., Cohen-or, D., Yeshurun, H.: Fragment-Basedimage completion. ACM Trans. Graph. 22(3), 303–312 (2003)
22. Cornea, D., Silver, D., Min, P.: Curve-skeleton properties, applications and algorithms. IEEE Transactions on Visualization and Computer Graphics 6(3), 81–91 (2006)
23. Shamir, A.: A Survey on Mesh Segmentation Techniques. Computer Graphics Forum 27(6), 1539–1556 (2008)
24. 27. Sorkine, O., Lipman, Y., Cohen-OR, D., Alexa, M., Rossl, C., Seidel, H. P.: Laplacian surface editing. In: Symposium on Geometry Processing, ACM SIGGRAPH/ Eurographics, pp. 179–188 (2004)
25. 28. Wang, J., Bhat, P., Colburn, A., Agrawala, M., Cohen, M.: Interactive Video Cutout. In: SIGGRAPH 2004 Conference Proceedings, pp. 124–131. (2004)
26. Li, Y., Gleicher, M., Xu, Y.Q., Shum, H.Y.: Stylizing motion with drawings. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 111–122 (2003)
27. Eitz, M., Sorkine, O., Alexa, M.: Sketch Based Image Deformation. In: Proceedings of Vision, Modeling and Visualization, pp. 135–142 (2007)