Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Stylizing Motion with Drawings

Yin Li,[1†] Michael Gleicher[2], Ying-Qing Xu[3], Heung-Yeung Shum[3]

[1] University of Science and Technology, Hong Kong
[2] University of Wisconsin, Madison
[3] Microsoft Research Asia, Beijing

Abstract

In this paper, we provide a method that injects the expressive shape deformations common in traditional 2D animation into an otherwise rigid 3D motion captured animation. We allow a traditional animator to modify frames in the rendered animation by redrawing the key features such as silhouette curves. These changes are then integrated into the animation. To perform this integration, we divide the changes into those that can be made by altering the skeletal animation, and those that must be made by altering the character's mesh geometry. To propagate mesh changes into other frames, we introduce a new image warping technique that takes into account the character's 3D structure. The resulting technique provides a system where an animator can inject stylization into 3D animation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Characters in traditional animation are remarkably expressive: they can change any aspect of their appearance in response to anything. Their appearance can depend on their normal shape, their pose, their mood, the direction they are being viewed from, or anything the animator desires. How they appear at one instant in time may be different from how they appear at any other time. Good animators can use this total control over characters' appearance to give characters remarkable amounts of personality and expressiveness, often exaggerating elements for emphasis or purposefully shaping the apparent silhouette to make the character easier to "read." The cost of this freedom is a burden on the animator: not only *can* all aspects of a character be controlled, they *must* be controlled.

Parametric 3D character animation offers a different balance of control and effort. For example, the animator may control the angle of an elbow, causing the arm to bend appropriately. With a well-designed model, muscles might bulge as the arm flexes. However, any changes in appearance must be driven by the parameters. The animator cannot adjust the appearance in response to the character's situation - for example to make the muscle weak in fear, or have a stronger silhouette shape so it is more clear - unless this type of control was specifically designed into that parameterization.

In motion captured animation, these problems are exacerbated. Motion capture data provides the overall pose of the character, a skinning model creates a 3D shape from this pose, and 3D viewing creates the final appearance. There is little opportunity for an animator to inject expression. Therefore, the resulting animation often lacks the appeal of traditional animation because the motion is copied from reality, and the shape and appearance of the character is not tuned to the specific needs of the moment of the animation.

Our goal is to inject some of the expressiveness of traditional 2D animation into motion captured 3D animation. To do this, we provide a tool that allows traditional 2D animators to use their talents to enhance the more realistic 3D animation. We give the animators total control over the characters' appearance at specific *key* instants, allowing them to create the situation specific effects that give traditional animation its distinctive style. We call the act of adding situation specific effects *stylization.*

To utilize animators traditional skills, we allow them to control the characters through 2D drawings. The disadvan-

† e-mail: liyin@ust.hk This work was done when Yin was working with Microsoft Research Asia for his internship.

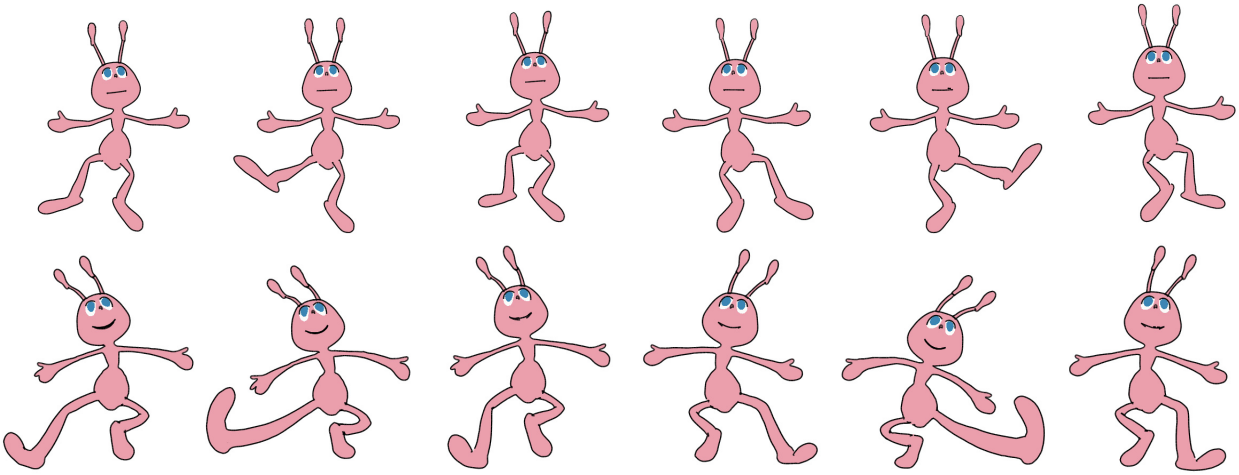


Figure 1: *Top: Original 3D model with motion capture data 2D. Bottom: Stylized animation driven by same motion capture data*

tage is that 2D drawings do not uniquely specify a 3D configuration. In order to integrate a drawing into an animation, we must guess an interpretation of the 2D sketch. This problem is ill-posed. Rather than trying to solve this impossible problem, we instead focus on decreasing the need for correct sketch interpretation.

A key technical challenge of our work is integrating the changes to nearby frames of the animation with smooth transitions. We need a warping method capable of handling large disparities with large amount of complexity. Our method decomposes the disparity into two parts: those that can be accomplished by altering the skeletal animation of the character, and those that must be accomplished by altering the positions of individual vertices of the character's surface. The skeletal changes can encode large amounts of change to the overall character and are easy to integrate into the animation, while detailed displacements are capable of representing arbitrarily detailed changes, but are more difficult to integrate into the animation. To address this latter issue, we provide a semi-automatic correspondence mechanism that makes use of hand-drawn images as well as 3D meshes, and a novel image warping method that accounts for the character's 3D structure.

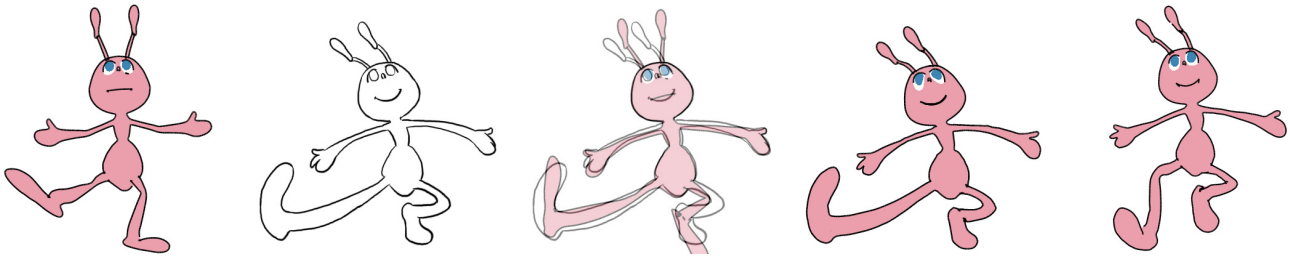
In this paper, we introduce our approach for empowering traditional animators to add personality to 3D motion captured animation. We begin by discussing an example, to illustrate the components of the technique as well as to articulate the workflow and interaction requirements. A survey of related work shows that existing approaches do not address our problem, but provide some building blocks for our work. In Section 3, we describe our methods for specifying the desired changes, transferring them to the 3D model, and propagating them into the animation. We conclude with example results and an evaluation of the limitations of our approach.

1.1. An Example: The Users Perspective

We introduce our approach from the user's perspective. Frames from the initial input and final results are shown in Figure 1. We begin with an animation created by applying a motion captured dance movement to a model of an ant character, as seen in the top of the figure. The "cartoony" character design rendering style suggests a cartoony motion. While the movements may be realistic if applied to a human[†], they are inappropriate for the character. Human movements seem stiff, conservative, and rigid in an otherwise cartoon animation.

To improve the animation, an animator selects frames to edit. In this example, an experienced 2D animator has chosen two frames where the ant is sticking out its leg (one to the left and one to the right). She then redraws the outline and key features of the character using the 2D drawing tools she is familiar with, as seen in Figure 2. The updated drawings are different from the original: she exaggerates the size of the extended leg to emphasize the kicking motion; she curves the bent leg to make it appear more flexible and to make the dip deeper and more energetic; she bends the antennae to make the character less rigid; she puts a happier facial expression on the character's face; she adds a finger to the character's outstretched hand; etc. In short, she uses her talents as an animator and her insight into the ant's personality and actions in a way that may never be automatable.

[†] It would be hard to call any movement of a smiling 4 legged ant realistic.



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

Figure 2: *(a) The motion captured movement is stiff and lacks personality. (b) The example image was drawn by the animator to better express the character's personality. (c) After motion editing, the posture is closer to the example image. (d) After layered warp, the mesh is warped to the example drawing's shape. (e) at another time, the warping field is propagated.*

Our system must now integrate her new drawings into the animation in a seamless manner. We call the new hand drawn images the *examples*, and the original computer drawing image sequence the *renderings*. Each example corresponds to one of the renderings. Because the example might have a considerably different appearance than its corresponding rendering, we consult a user (who need not be the animator) to help specify the connection between the two.

Our approach divides the differences into two parts: those that can be achieved within the framework of the 3D skeletal animation, and those that cannot. We first attempt to account for the differences between the rendering and example by altering the 3D skeletal animation. We alter the pose of the skeleton, adjusting its joint angles and segment lengths so that the rendered character appears to match the example drawing as closely as possible. At present, this is an interactive process, where the example image is overlayed on top of a 3D view of the character and the user can employ forward and inverse kinematics to adjust the pose. Views of this process can be seen in Figure 4.

Adjusting the pose of the 3D skeleton cannot achieve all of the differences between the original rendering and the example image. Our system requires the user to specify a small number of corresponding points on each, as shown in Figure 5. The system then generates a seamless transition between the original rendered animation and the example frame. This transition occurs over a user-specified duration of time (usually a fraction of a second). The result of the process is a new animation that is similar to the original, but contains the example image.

Our process is optimized for characters that are created by driving a skin from an underlying skeleton using blend-based skinning. Such skinning methods are supported by all animation systems, and most interactive runtime engines. Our system uses the skinning and skeletal motion information in creating the transitions between the rendered and example images.

1.2. Technical Overview

There are three key insights behind our work:

1. The differences between the initial animation and example sketch can be decomposed into two parts, those that can be created by altering the character's skeletal motion, and those that cannot. This insight allows us to provide the flexibility of arbitrary changes, with the ability to make large changes in the overall motion. Changes to the skeletal motion are easier to blend and interpolate, and the image-plane mesh deformations (which have no information about depth) are most likely to work best on smaller disparities (see Figure 3.) It is important to accomplish as much of the changes with 3D motion editing as possible.
2. The 3D skeletal animation can provide a good model for the changes that occur in the images. This insight is important since it allows us to devise methods that preserve the discontinuous nature of the image and the 3D motion of the character when working with a 2D image.
3. The 2D deformation field can be viewed as a displacement field along the surface of the 3D character, and therefore propagated by the motion of the character. This is important as it provides a way to propagate changes along the animation.

We call our method for using the 3D skeletal animation information to drive the image deformations *multi-layer morphing*, and use it in three different steps in our approach. Such a technique is necessary because a conventional image morphing method that does not utilize information about the character would exhibit artifacts where there are nearby or overlapping parts of the character, and they would not provide any method for using the motion of the 3D character to find compute other images.

Our approach has the following steps:

1. The user adjusts the skeletal pose that corresponds with the example image, and the system creates a new mo-

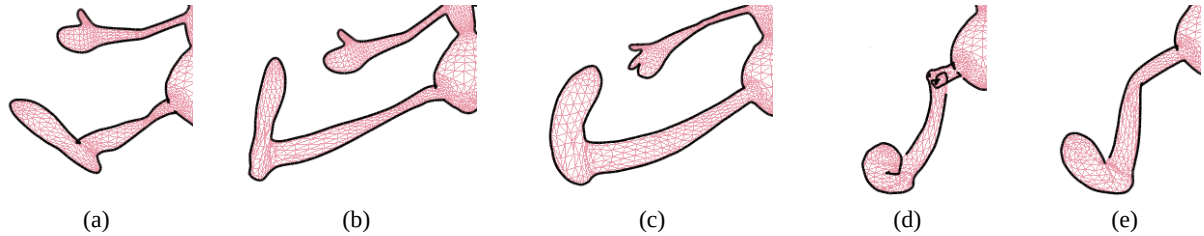


Figure 3: *Importance of motion editing before warping. Both the original 3D mesh(a) and the mesh after motion editing (b) can be warped to (c). However, on another time t, warping the original mesh will cause errors (d). Once the motion editing compensates for the difference, the warping gives a better result (e).*

tion (and corresponding set of renderings) that interpolates this pose using motion warping (Section 3.1).
2. The user provides a small set of correspondences between the rendering and the example image and the system expands these points into curves (Section 3.2).
3. The system creates a dense image warp between the example and rendering using an interpolation technique based on the 3D structure of the model. This associates pixels in the example image with the 3D data (Section 3.4).
4. The system interprets the image-space field of the image warp as a vector field along the surface of the character (Section 3.5).
5. The character's appearance is computed at times other than the example by driving the vector field along with the motion of the character (Section 3.6).
6. The system fades these changes in and out over time (Section 3.7).

2. Related Work

The ability of traditional (2D) animators to create expressive characters was developed early in the history of the art form. In computer animation, support for this expressive style has primarily been in the form of 2D tools such as TicTacToon [6] or Toon Boom [24]. Bregler's efforts to capture this expressive movement [2] presents methods that capture both shape deformations as well as rigid movements, as 2D animators make extensive use of both. Unlike this work, we will allow 2D animators to enhance 3D animations, rather than trying to capture existing 2D animations.

Traditional animators make use of the fact that the personality of a character or movement can be expressed by having the character have a specific appearance when in a characteristic pose. In fact, the standard way of documenting a character is to show these characteristic poses in what is commonly known as a *model sheet* [23 9]. This concept inspires our work to permit animators to stylize animations by specifying the appearance of the characters at characteristic poses.

The use of these characteristic poses is the notion that the appearance of a character depends on the situation, including the character's pose and the viewing direction. In the real world, or in standard 3D graphics, the appearance is based on an underlying 3D shape. Rademacher's View Dependent Geometry (VDG) [14] makes the character's appearance depend on the viewing direction, and Lewis et. al's Pose-Space Deformations (PSD) makes the appearance depend on pose [11]. We make the appearance depend on the situation. Like Rademacher [14] we allow the user to control the appearance directly in the view, although we provide tools that make this easier for the user. Also like VDG, we interpret image deformations to find a mesh deformation that may only be correct near the given viewpoint. Like PSD and its variants [17], we keep the changes relative to the character's underlying skeleton so that they move correctly.

Rademacher's VDG approach [14] could be used to stylize motion as we do, however with more effort. Rademacher requires his users to use a 3D modeling tool to adjust the model in 3D, vertex by vertex, whereas we allow an artist to use 2D drawings and a small amount of intervention to provide correspondences. Rademacher also does not consider how to use the 3D skeletal structure of a character to better propagate the changes between example poses and infer about occluded regions.

Corrêa et al [5] also connect drawings from animators to simpler 3D models. They perform a 2D warp of the model in screen space in order to draw textures from the model into the drawing. Similar to our approach, the user specifies correspondences between the drawing and silhouettes of the model. Unlike our approach, they warp the model in 2D. Also, our approach specifically considers articulated figures, and is able to propagate the changes on one frame to another frame.

Terzopoulos and Witkin [21] simulated models by dividing the motion between into that which could be represented simply (as a rigid body in their case), and that which required detailed deformation of the shape. Like many others since, we also divide motion into that which can be represented simply (as a linear blend skin in our case) and that which must be represented as detailed deformations.

The problem of interpolating a set of finite controls to produce a continuous field is known as scattered data interpolation, and is an important problem for image warping. A survey of the basic issues involved is provided by Wolberg [27]. Beier and Neely considered using curves as the controls [1], although we actually treat curve controls as a set of points. To create the interpolation field over points, we use the Multilevel Free Form Deformation technique of Lee et al. [10].

Interpolation or warping methods work better when and appropriate model for the field is used. For example, View Morphing [16] uses a perspectively correct model to find motion fields that mimic object motions. For our model, we use the articulated figure motion. Ju and Black [8] introduced a pixel motion model based on a 2D articulated figure, and Bregler and Malik [3] provided image flow based on a 3D articulated model. In both cases, the pixel flow model was used for tracking, assumed a completely rigid figure, and did not account for significant deviations from the model. Therefore, we create the multi-layed methods of Section 3.4.

Many in the computer vision community have tried to determine the shape of 3D objects from silhouettes, or more general line drawings. For example, Terzopoulos et al. [22] find the most symmetric shape that fits a given silhouette, Malik and Maydan discussed recovering shapes from curved silhouettes [12], and Szeliski has presented several methods for find the shape of the object from multiple silhouettes [18, 19]. We do not claim that the method we have chosen is either novel or superior. It is, however, simple and sufficient for our application.

3. Injecting Style into Animation

As described in Section 1.1 the process begins with a segment of animation created from motion captured data. Such motion is stored as a sequence of skeletal poses, where each pose consists of the position and orientation of the character's root, the angles of each of the character's joints, and the lengths of each of the segments of the character's rigid skeleton. We will refer to the initial motion captured data as $\mathbf{q}^0$, with an individual pose at time t being $\mathbf{q}^0(t)$.

We will use uppercase letters to denote images, with R meaning an image that was rendered from the 3D character. Therefore $R^0(t)$ denotes the initial animation, the frames rendered from the initial motion data. We will denote the example image (drawn by the animator) as X. The example image applies at a specific time in the animation, that we denote as t_e, so that X corresponds to $R^0(t_e)$. For symmetry in notation, we will also refer to the drawn example as $X(t_e)$ as a reminder of the time at which it applies.

We will denote mappings or fields by Calligraphic letters. For example, a warping field $\mathcal{W}_{A,B}$ specifies a point in image B that is associated with each point in image A. Similarly, a disparity field $\mathcal{D}_{A,B}$ would specify a vector for each point in A that leads to its corresponding location in B, or

$$\mathcal{D}_{A,B}(x,y) = \mathcal{W}_{A,B}(x,y) - (x,y).$$

When an example frame is chosen, the user must also specify an duration for the example to effect. We call this duration of time the effective range of the change. The range of frames is always centered around t_e and has "radius" Δ so that the range of frames effected is from $t_e - \Delta_b$ to $t_e + \Delta_a$ exclusive. While normally, the example frame is in the center of the changed duration, we allow for different timings before and after the example.

3.1. Motion Editing

As described in Section 1.1, an initial step in our approach is to edit the skeletal animation such that its approximates the example image. We refer to the resulting motion as $\mathbf{q}(t)$, and the rendered images created from this motion as $R(t)$. The task of motion editing is to choose $\mathbf{q}(t)$ such that $R(t_e) \approx X$, while making as visually insignificant a change to the motion overall.

Our motion editing process first determines the pose, $\mathbf{q}(t_e)$, that causes the resulting rendering to be as similar to the example image as possible. This is done by adjusting the position of the root, the angles of the joints, the length of the skeletal segments, and the scaling of the geometry in each of the bones' coordinate systems. At present, this is done interactively by overlaying the example image over the rendering and allowing the user to adjust the parameters by a combination of forward and inverse kinematics. An example is shown in Figure 4.

While it may be possible to automate the choice of $\mathbf{q}(t_e)$ by performing an optimization on the residual error $X(t_e) - R(t_e)$, we have chosen not to implement this because the amount of user intervention required to perform this step

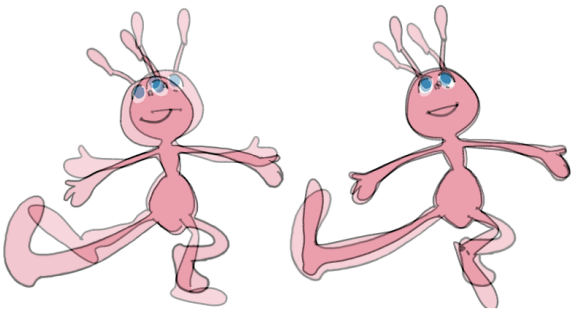


Figure 4: *Motion editing is performed by overlaying the example image over the rendered frame (left) and adjusting the character's skeletal parameters until the images match (right).*

(given an effective user interface) is quite minimal. Also, existing methods for performing these types of pose optimization, such as [15] or [3] or [20] would need to be extended to account for the large amounts of shape deformation that cannot be accounted for by changing the skeleton.

To create $\mathbf{q}(t)$, we use a motion warp [26], also known as a motion displacement map [4]. That is, we compute

$$\mathbf{q}(t) = \mathbf{q^0}(t) + \mathbf{d}(t),$$

where $\mathbf{d}(t)$ is a specially chosen *displacement map*. We construct $\mathbf{d}(t)$ as a pair of cubic Hermite segments, one over time range $t_e - \Delta_b$ to t_e, and the other from t_e to $t_e + \Delta_a$. We set the controls of these splines such that

$$\mathbf{d}(t_e) = \mathbf{q}(t_e) - \mathbf{q^0}(t_e)$$
$$\mathbf{d}(t_e - \Delta) = \mathbf{d}t_e + \Delta = \mathbf{0}$$

and that the first derivative of $\mathbf{d}(t)$ at these three points is zero.

3.2. Correspondence Specification

The motion editing will bring the features of the rendered image closer to the drawn example. We next need to find a warping field, $\mathcal{D}_{R(t_e),X}$, that encodes the remaining disparity. We do this as a two step process: first we find correspondences between all of the drawn features in X and corresponding points in $R(t_e)$; second, we interpolate these correspondences across the entire image plane to achieve a dense field.

The large potential differences between X and $R(t_e)$ makes automatic correspondence difficult. To provide a reliable method, we allow the user to provide some matching points. User intervention also allows for creative choice in feature matching, as seen in the choice of where the finger emerges from in Figure 5.

Corresponding the feature curves well may require a large number of points, so manual specification is prohibitive. Specifying this large number of corresponding manually would be prohibitive, so we provide a semi-automatic approach. We denote correspondence pair i as C_i. It consists of a 2D position $\mathbf{x}_i$ in image X, and a point $\mathbf{r}_i$ in image $R(t_e)$. All of the points $\mathbf{r}_i$ must either be vertices of the character's mesh, or points on the edges between two vertices.

Our correspondence approach begins by automatically computing the silhouette and creases of the mesh model that is rendered in $R(t_e)$. These edges are most likely to create strong features in the rendering, and, therefore are most likely to need a correspondence. We detect the silhouette from the 3D mesh using the technique introduced by Markosian et al [13]. This method produces a long chain of points resulting in a smooth silhouette curve, avoiding the zig-zagging common in simpler approaches. If the mesh is not dense, we subdivide mesh to improve the silhouette appearance. To further enhance the quality of the computed silhouette, we filter the mesh to avoid ruffles, by shrinking the vertex to the average of its neighboring vertices. To compensate for volume that is lost by the filtering, we perform a re-inflate the character by displacing each vertex along its outward-facing normal.

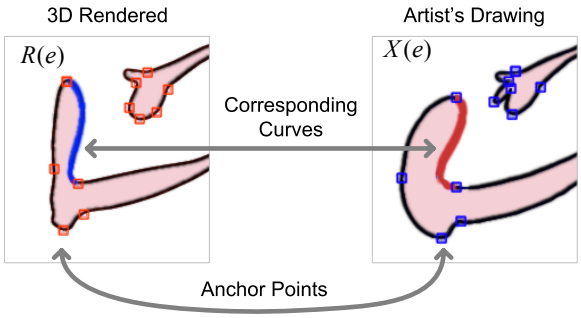


Figure 5: *UI at time t_e. The user places a few anchor points (highlighted by squares), snapping them on corresponding positions in the example image. The corresponding curves in between these anchors are computed using a snake operator.*

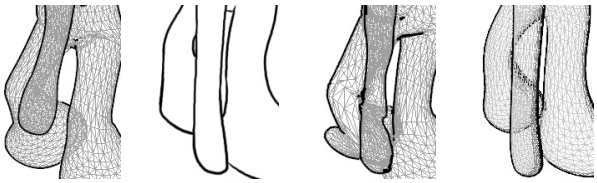
To specify a corresponding pair of curves, the user selects a few points on the silhouette of $R(t_e)$, and specifies the corresponding locations in X. The system connects the points by following silhouette edges in the mesh model to find a curves in the rendering, and applies a snake operator to find an image edge that connect points in the example image, as seen in Figure 5.

3.3. Skinning

Our needs require a pixel flow model that works for morphing, handles non-rigid blend-based skinning, and accounts for significant off-model effects, leading to the development of the warping method detailed in this paper. A key idea in our image warping method is to make use of the 3D skinned model that is imaged in the rendering. Therefore, before explaining our method in detail, we review the blend-based skinning method, sometimes referred to as skeletal subspace deformation (SSD) [11].

Blend-based skinning begins with a model of the character in what is referred to as *dressing pose*, and deforms this model based on the skeletal pose at a given time. We denote the positions of vertex j in the dressing pose as $\mathbf{v^d}_j$. For each vertex, we specify a skin weight vector $\mathbf{w_j}$ whose entries control how much each bone affects the vertex (e.g. $\mathbf{w}_{ji}$ specified how much bone i affects vertex j). A resulting vertex position is computed for animation time t by

$$\mathbf{v}_j(t) = \sum_i \mathbf{w}_{ji}\mathbf{M}_i(t)\mathbf{v^d}_j, \qquad (1)$$



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

Figure 6: *(a) The original mesh to be warped; (b)The artist's sketch;(c) Warping the mesh without layering information: the morphing field confuses independent regions. (d) With the layering information, the warping is cleaner and smoother.*

where $\mathbf{M}_i(t)$ represents the transformation associated with bone i at time t, relative to the bone's dressing pose.

3.4. Multi-Layer Image Warping

The previous section gives us a set of curve correspondences between X and $R(t_e)$. With this, the disparity field over the image $\mathcal{D}_{R(t_e),X}$, is known at the points along the curve, but we need to perform scatted data interpolation to fill in the rest of the image. The scattered data interpolation techniques used in image warping create a smooth warp field to avoid introducing visual discontinuities. Unfortunately, such smooth warping fields are inappropriate for articulated figures where two points may appear close together in the image but actually be on different parts of the character at different depths. This is particularly problematic when the character exhibits self-occlusion. In order to avoid the artifacts shown in Figure 6, we introduce a multi-layer warping method.

Our layered morphing technique is based on the idea that the different segments of the articulated figure move differently, and therefore their images should be warped differently. We therefore allow each bone to define its own warping field as an independent layer. Because every correspondence is associated with a mesh vertex, it is also associated with a skin weighting vector. We use the skin weighting vectors ($\mathbf{w}_j$) to determine layers as well.

For bone i, we define its warping field $\mathcal{W}_i$ by using the correspondences whose vertices are influenced by the bone. We use the multi-layer free form deformation (MFFD) warping method [10] for each layer, because it insures that the resulting warp does not have any folding in image plane. We weight the influence of each correspondence to $\mathcal{W}_i$ by the vertex's weight in bone i.

To compute the warp of a point in $R(t_e)$, we use a weighted linear combination of the different layer warps. The weights are taken from the skin weights associated with the vertex. To draw the warped version of $R(t_e)$, we compute the image location of each vertex, use its skin weight vector to determine how to blend the warp layers, and then connect these vertices with triangles after performing the warping. By drawing the triangles after warping the vertices, the cracking issues possible with forward map image warps are avoided.

3.5. Interpreting the Warping Field

The previous section provides a disparity field $\mathcal{D}_{R(t_e),X}$ between the images $R(t_e)$ and X. Because we only observe this field (or, to be more precise, its effects) in the image plane and for visible points on the character, it is impossible to know completely its 3D values, or how it behaves on the occluded parts of the character. There are many possible 3D shapes that might lead to the appearance of the example image, and we have little information to decide amongst them.

If we were only viewing from this one direction, this incomplete data would not matter - we would not see it. However, as the character moves, occluded regions may become visible, and parts may rotate so that what once was parallel to the viewing direction may now be visible. Therefore, we must make some guess as to the 3D shape behind the example.

Fortunately, the nature of our guess is not too critical because we will only be using the guess near the pose where the missing information is invisible, and because the disparities are small because we have used the motion editing for the largest part of the overall difference. Therefore, we choose a simple approach for interpreting the disparity field as a 3D vector field over the surfaces of the character's geometry. We attach these vectors to the skin so that they move along with the character.

We define the 3D disparity field $\mathcal{M}$ over the surface of the character's mesh (or, equivalently, the warping field), as

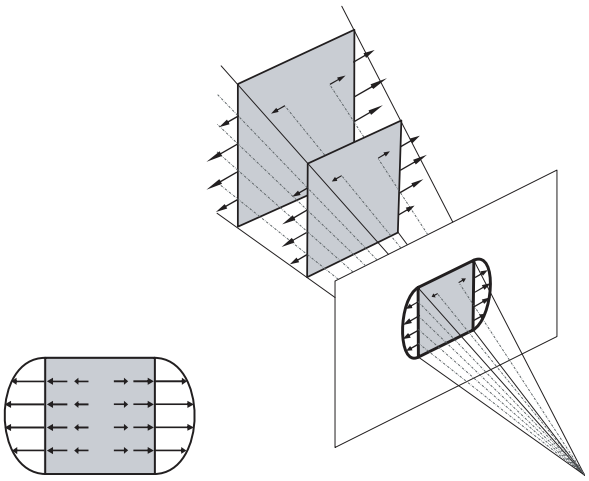


Figure 7: *Interpretation of the 2D disparity field as a 3D field along the surface of the mesh.*

attaching to each 3D point a 3D vector that is parallel to the image plane. These vectors are chosen such that their projection is the same as their value in the 2D disparity field, as shown in Figure 7.

The interpreted disparity field provides a 3D model. While this model could potentially be viewed from any direction, it will only have the correct appearance in the view that the animator specified.

3.6. Warp Field Propagation

The warping field provided by the previous section could be faded in and out to provide a smooth morph between the example image and the rendered image. This could be done in either the image plane, or in 3D. However, our goal is not to morph between the static images, but instead to integrate the example into the motion of the animation. When $t \neq t_e$, we do not want to warp between $R(t)$ and $X(t_e)$. Instead, we must approximate $X(t)$. Fortunately, since we are only using $X(t)$ for morphing, its contribution becomes small as t gets far from t_e, so it is less important that our approximation of $X(t)$ is perfect far from t_e.

To approximate the example over time, we assume that the displacement field is attached to the character in 3D, rather than simply in the image. As the points of the character move, the attached vectors move similarly.

The motion of a vertex on the mesh is given by the skinning equation in Equation 1. The transformation for a vertex j from time t_e to time t is therefore

$$\mathbf{T}_j(t) = \left(\sum_i \mathbf{w_{ji}}\mathbf{M}_i(t)\right)\left(\sum_i \mathbf{w_{ji}}\mathbf{M}_i(t_e)\right)^{-1}. \qquad (2)$$

The motion of the field with respect to the surface that it is attached to is not necessarily given by the attached point. There are two obvious choices: the vector should move along with the surface, as the tangent vectors do; or the vector should move so that it preserves its relationship with the surface, as normal vectors do [25]. In practice, these two are the same when the matrices are isotropic (that is only have uniform scales and no skewing), which the matrices produced by Equation 2 are always at least close to being. We see little difference between using $\mathbf{T}^R = (\mathbf{T}^{-1})^T$ or $\mathbf{T}^R = \mathbf{T}$.

We can compute the disparity field for any point on the character at any time

$$\mathcal{M}(t, \mathbf{v}_j) = \mathbf{T}_j^R(t)\mathcal{M}(t_e, \mathbf{v}_j). \qquad (3)$$

We note that because we have limited information about the appearance of "other sides" of the example, simulating its motion by simply attaching the morphing field on a 3D mesh is a crude approximation of what the real motion might be except very close to t_e. In the next section, we will see why this is not crucial.

3.7. Assembling the Final Animation

Given the ability to compute the deformation to the character at any time, we can apply these changes to produce a final animation. We define a blending function $\alpha(t)$ with value 1 when $t = t_e$, and value 0 when $t = t_e - \Delta_b$ or $t = t_e + \Delta_a$. The resulting character mesh is therefore

$$\mathbf{v}_i^S(t) = \mathbf{v}_i(t) + \alpha(t)\mathcal{M}(t, \mathbf{v}_i).$$

This equation makes clear that the quality of the warping field is only important when t is close to t_e, as $\alpha(t)$ will be small otherwise.

If there are multiple example images, each will create its own displacement field and have its own blending function. These additional terms can be added into Equation 3 as needed.

4. Results

We have implemented the stylization approach by integrating it into two separate systems. Motion editing is done in our character animation testbed, and a separate system handles the correspondence, interpretation, and geometry alterations. The systems communicate through files. Both systems include stylized renderers that produce the cartoon rendering style that we prefer for our experiments and run on personal computers. The 2D drawings were created with a 2D display tablet and by drawing on paper and scanning.

Many of the examples were created by a professional 2D animator with no experience with 3D animation, but excellent drawing skills. Other examples were created by computer scientists. The task of creating drawings is not too difficult for the untrained artists as the original rendering can be used as a guide for tracing.

The dancing ant animation in Figures 1 and 2 was created by applying a standard library motion to a standard library character (both available freely on the web), using a motion retreating technique [7]. We took a short loop of dancing motion and showed it to a professional animator who redrew two frames. Figure 8 shows a character from a commercial database applied to a different stock motion and stylized.

Figure 8: *Another example of our system. From left to right: Original frame, the sketch, the result frame, another frame propagated.*

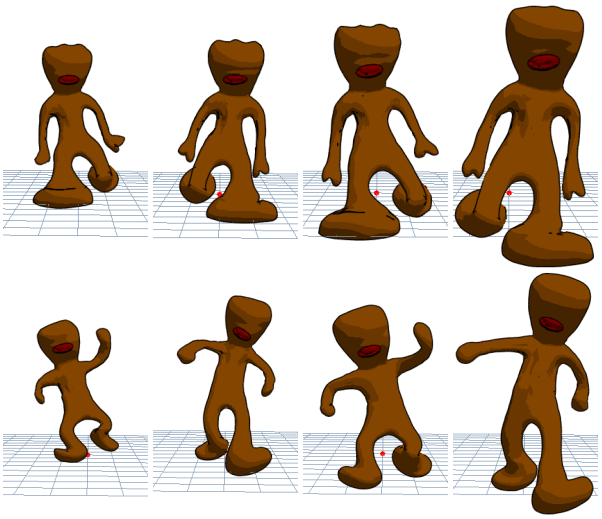


Figure 9: *Top: walk stylized by amateur. Bottom: stylized by animator*

The quality and character of the result is very dependent on the skill of the animator making the drawings. In Figure 9 shows an example stylized by two people: a computer scientist and a professional animator. The images drawn by the former were intended to stress the system's features more, and are detailed in Figure 10.

Figure 11 shows frames of a stylized fight between two characters. Here, the animator chose a few frames to exaggerate. The stylization were done together so that the motions still fit. Many of the effects are specific to the situation, for example, the way the characters stretch and twist in response to the punches and kicks.

5. Conclusion

Our stylization method allows us to use an animator's talents to add stylization to otherwise realistic motion capture animation. We allow the animator to use their 2D drawing skills, and integrate their sketches into the original animations. Because we have chosen to allow animators to work with 2D drawings, our method is limited: changes made to the key frames are unlikely to be applicable elsewhere. We plan to have our system automatically identify places in animations where examples are appropriate, adjust them so that they fit, and apply them as presented in this paper.

The primary contribution of our work is to provide an overall method for stylizing motion captured animation. While similar results might have been achievable with other methods, such as Rademacher's View Dependent Geometry [14], we contribute several techniques that allow the deformations to be specified as traditional drawings, and to work better with articulated characters. These techniques include semi-automatic correspondence to minimize the user's effort, and multi-layer warping to use the articulated structure of the 3D character to avoid warping artifacts and provide a method that can work in the face of self-occlusion.

One fundamental limitation of our technique is that we only consider alterations of the key frames. Much of a motions style can come from its timing and from the trajectories of parts of the character. Our approach offers no way to adjust these time-related characteristics. We plan to integrate our work with some scheme for making time adjustments.

Another limitation of our technique is that features of the characters that are not specified by the animator may not move in appealing ways. In practice, this means that our methods work best with characters that have little detail beyond their silhouettes. Stylizing the motion of more detailed characters requires more input from the animator, and might require extensions to our interface to allow specifying features that are not creases and silhouettes.

While our method is successful on the examples we have tried, we believe that some extensions would lead to improved results and decreased effort. For example, better motion editing tools that helped the user match the character's pose to the images would reduce the need for 3D animation experience in the process. A better approach for interpreting the 3D field, possibly by performing the interpolation over the surface of the character, might improve the robustness and appearance of the results. A key failure in our approach is that it requires the character's model to have enough flexibility to express the changes desired by the animator. Presently, this requires us to use densely tesselated character meshes. In the future, we hope to address this with dynamic tessellation.

In general, we have found the limiting factor in creating examples to be creative, not technical. Our method allows us to use our creativity in how characters appear, to express these ideas as 2D drawings, and to integrate these drawings into motion captured animation. We divide the changes required to realize the drawings into those that can be made by adjusting the 3D skeletal animation, and those that cannot. The former are handled using motion editing, while the latter are handled by a combination of scattered data interpolation, 3D interpretation, and vector field propagation. Together, they yield animations that incorporate an animators 2D expression into motion capture data.

Acknowledgement

We would like to thank animator Yuanyuan Su for providing the drawings used in the examples. We would like to thank Taylor Wilson and Tom Tolles of House of Moves Studios for providing motion capture data. Work in Wisconsin was supported in part by NSF grants CCR-9984506, CCR-0204372 and IIS-0097456 and gifts from Microsoft and Intel. We would also like to thank Dr. Xin Tong of MSR Asia and members of the UW graphics group for their help with this project.

References

1. Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, volume 26, pages 35–42, July 1992.
2. Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics*, 21(3):399–407, July 2002.
3. Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 1998.
4. Armin Bruderlin and Lance Williams. Motion signal processing. In *Proceedings of SIGGRAPH 95*, pages 97–104, August 1995.
5. Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 435–446, July 1998.
6. Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas, and Frédéric Taillefer. Tictactoon: A paperless system for professional 2-d animation. In *Proceedings of SIGGRAPH 95*, pages 79–90, August 1995.
7. Michael Gleicher. Retargeting motion to new characters. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, pages 33–42. Addison Wesley, July 1998.
8. S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated motion. In *2nd Int. Conf. on Automatic Face- and Gesture-Recognition*, pages 38–44, Killington, Vermont, Oct 1996.
9. Larry Lauria. Larry's toon institute: Character model sheets. web page, 1999.
10. Seung-Yong Lee, Kyung-Yong Chwa, and Sung Yong Shin. Image metamorphosis using snakes and free-form deformations. *Computer Graphics*, 29:439–448, 1995.
11. J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, pages 165–172, July 2000.
12. Jitendra Malik and Dror Maydan. Recovering three dimensional shape from a single image of curved objects. *IEEE Trans on PAMI*, 11(6):555–566, June 1989. 1989.
13. Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. In *Proceedings of SIGGRAPH 97*, pages 415–420, August 1997.
14. Paul Rademacher. View-dependent geometry. In *Proceedings of SIGGRAPH 99*, pages 439–446, August 1999.
15. J. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proc. 1995 IEEE Intl Conf. on Computer Vision*, Cambridge, MA, 1995.
16. Steven M. Seitz and Charles R. Dyer. View morphing: Synthesizing 3d metamorphoses using image transforms. In *Proceedings of SIGGRAPH 96*, pages 21–30, August 1996.
17. Peter-Pike J. Sloan, III Charles F. Rose, and Michael F. Cohen. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, 2001.
18. Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.
19. Richard Szeliski and Richard Weiss. Robust shape recovery from occluding contours using a linear smoother. *Intl Journal of Computer Vision*, 28(1):1998, 1998.
20. Camillo J. Taylor. Reconstruction of articulated objects from point correspondences in a single image. In *In IEEE CVPR*, June 2000.
21. Demetri Terzopoulos and Andew Witkin. Physically-based roodels with rigid and deformable components. *IEEE CG&A*, 8(6):41–51, November 1988.
22. Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models and 3d object reconstruction. *Intl Journal of Computer Vision*, 1(3):211–221, 1987.
23. Frank Thomas and Olliver Johnson. *Disney Animation: The Illusion of Life*. Abbeville Press, 1984.
24. Toon Boom Technologies. Toon boom studio. Computer Software, 2002.
25. Kenneth Turkowski. Transformations of surface normal vectors. Technical Report 22, Apple Computer, July 1990.
26. Andrew P. Witkin and Zoran Popović. Motion warping. In *Proceedings of SIGGRAPH 95*, pages 105–108, August 1995.
27. George Wolberg. *Digital Image Warping*. IEEE Press, 1990.

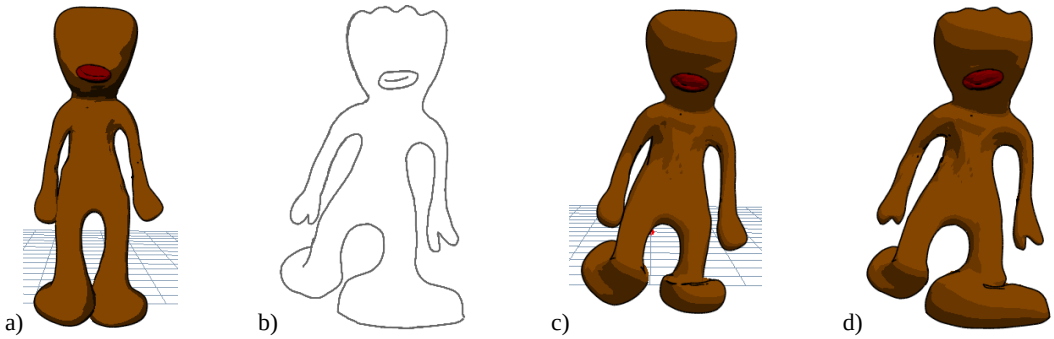


Figure 10: *Editing a walking motion (a) original frame (b) sketch, (c) motion editing only (d) final frame*

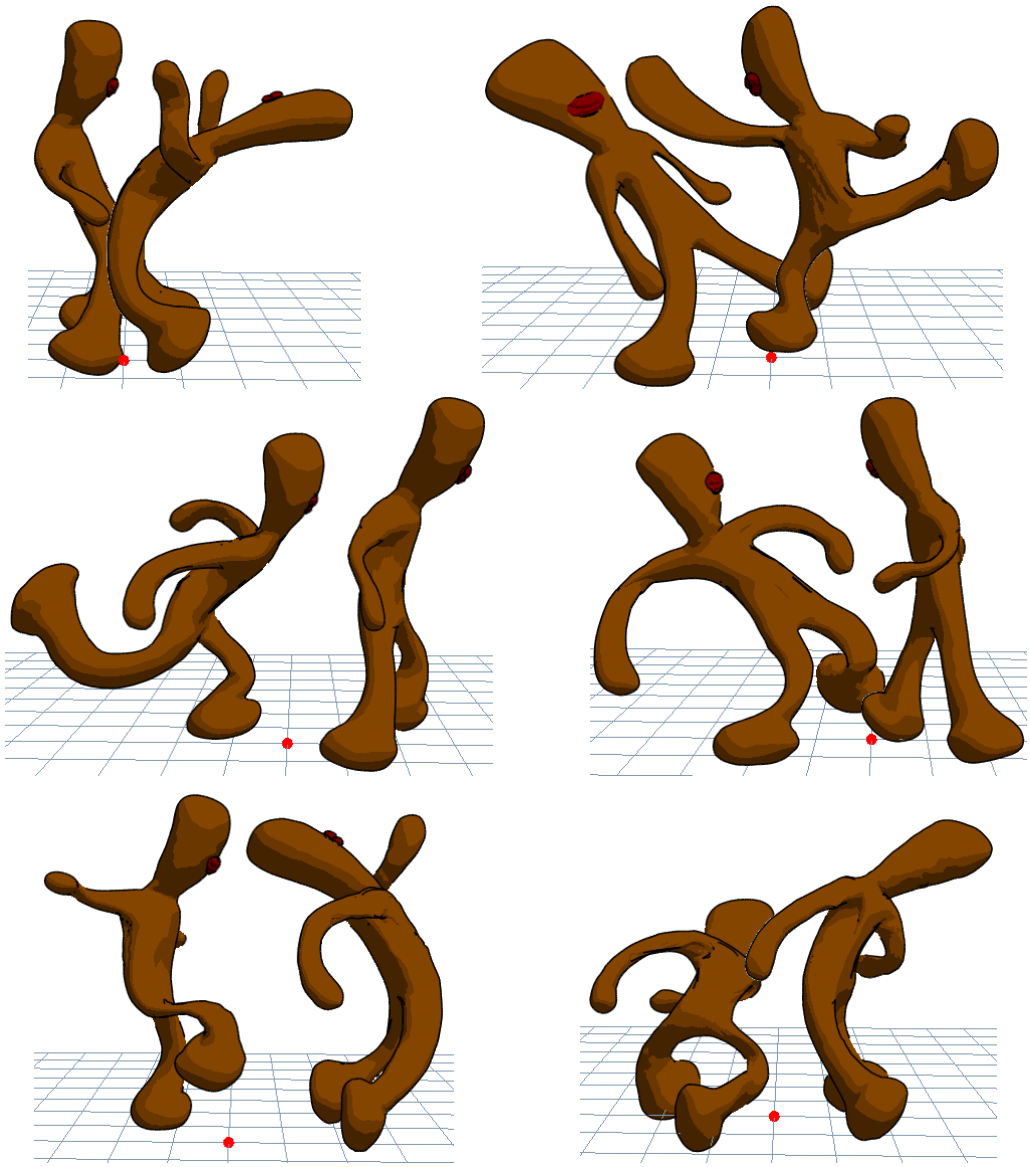


Figure 11: *A fight between two characters is stylized.*