Turning to the Masters: Motion Capturing Cartoons*

Christoph Bregler Lorie Loeb Erika Chuang Hrishi Deshpande
Stanford University†

Abstract

In this paper, we present a technique we call "cartoon capture and retargeting" which we use to track the motion from traditionally animated cartoons and retarget it onto 3-D models, 2-D drawings, and photographs. By using animation as the source, we can produce new animations that are expressive, exaggerated or non-realistic.

Cartoon capture transforms a digitized cartoon into a cartoon motion representation. Using a combination of affine transformation and key-shape interpolation, cartoon capture tracks non-rigid shape changes in cartoon layers. Cartoon retargeting translates this information into different output media. The result is an animation with a new look but with the movement of the original cartoon.

Keywords: Animation, Computer Vision, Deformations, Morphing, Object Tracking, Shape Blending, Video.

1 Introduction

In this paper, we present techniques to extract motion from traditional animation, a process called cartoon capture, and reuse these motions for other characters, a process called cartoon retargeting.

1.1 Motivation

We can think of animation as having two dimensions: the visual style (how the image looks, how it is rendered, the style of the drawing or model) and the motion style (how the characters move, the amount of exaggeration, use of cartoon physics and way in which the animation principles are used). The visual style of an animation can be anything from photo-realistic to abstract. The motion style also varies from one animation to another. It can range from robotic, to realistic to highly expressive (Figure 1).

Visual style and motion style usually go together. Most traditional animated cartoons have highly exaggerated drawings and highly exaggerated motions. These animations are usually created by highly trained animators who use animation principles to create motion that is expressive and stylized.

In contrast, photo-realistic computer animations are generally animated with realistic motion. Physical simulation and motion capture have been fairly effective for creating such realistic motion. Simulation techniques are mainly used to create low-level physical phenomena and motion capture is used to create realistic human motions (Green area in Figure 1).

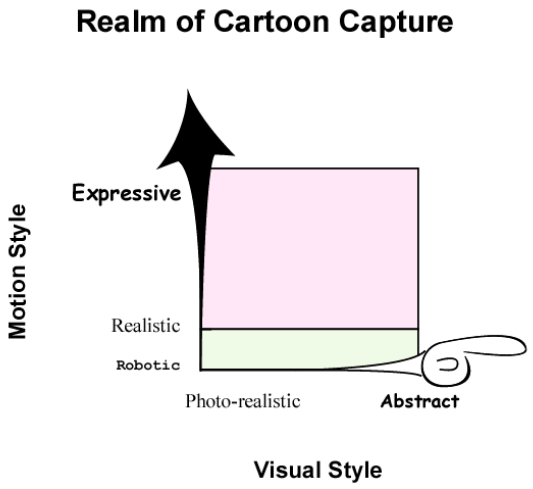

Figure 1: Animation Styles: Animation has both a visual style and a motion style. Motion capture works within the green area of this figure. The pink area represents the realm of cartoon capture.

Many research efforts have been devoted recently on how to reuse motion capture for different animation domains, how to adapt motion capture to different characters, how to retarget to different activities, and how to apply machine learning techniques to animation. In the game and special effects industry, motion capture is increasingly used to efficiently create realistic motion for movies and video games. Furthermore, much progress has been made in improving motion capture technologies. Higher accuracy, real-time and user-friendly systems, and the increased availability of large public motion capture databases and service bureaus, give even more reasons to use motion capture data for many kinds of character animation.

In contrast, stylized and expressive animation, as done by traditional animators, is seen as time consuming or even impossible. Skilled artists are rare and the costs are often prohibitive. Currently, traditionally drawn animations cannot be easily reused or transfered to different domains and characters, as is possible with motion capture.

The main point of this paper is to present new techniques that bridge the gap between those two worlds. Computer animation techniques using motion capture input are limited to a realistic motion style. Cartoon capture is able to transfer the cartoon motion style into a representation that can be used in the same fashion as standard motion capture. It can be seen as a new "front-end" whenever a more expressive and stylized motion style is needed. We turn to the masters – highly trained traditional animators – in order to find the best examples of expressive motion.

Example applications that call for expressive and stylized motions include game development, consumer level applications, feature film production, and general research on motion styles.

*http://graphics.stanford.edu/projects/tooncap

†bregler,lorie,echuang,hrshi@graphics.stanford.edu

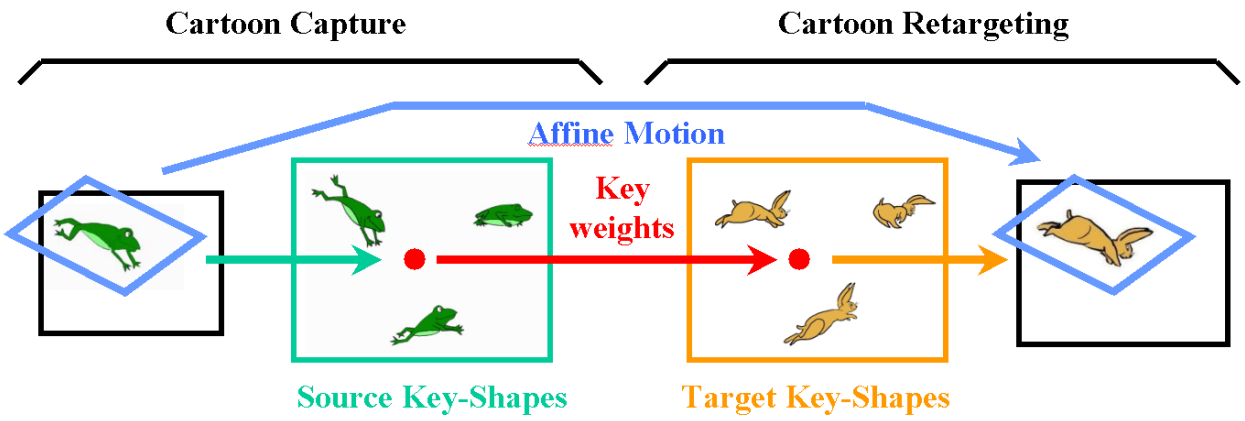

Figure 2: Overview chart of cartoon capture and retargeting.

Often traditional animators can create the desired motion style much quicker with pencil test drawings, than with tweaking 3D parameters in a computer animation program. This technique allows them to animate on paper, and then transfer the motion to a 2D or 3D character.

1.2 Scope

The pink area in Figure 1, illustrates the range of experiments we are conducting with this paper. We demonstrate how we can isolate the motion style of cartoon animation (upper part of Y-axis) and apply it to different visual styles (the entire range on the X-axis).

One future direction of this research is to qualify and quantify that pink area. As we better understand the differences between realistic motion and expressive cartoon motion, we can create filters that could be used in conjunction with motion capture to intensify or change the style of the movement. Examples include filters that could add squash and stretch, exaggeration or anticipation and follow-through. This could be used to create charicatures of famous people, or simply to allow more variety of motion from a single motion capture shot.

1.3 Challenges

Because we begin with a 2-dimensional animated video, existing motion capture techniques are not adequate. There are new challenges that need to be addressed:

1. Cartoon characters have no markers. Conventional tracking techniques that rely on point features cannot be applied here.

2. The low frame rate makes tracking difficult. Typical motion capture systems sample at 60-200 frames per second, while animation is usually recorded at 24 frames per second. Each image is often held for 2 frames, thus using only 12 images per second. This makes the change between images relatively large.

3. Identifying limb locations in cartoons is difficult. Also, cartoon objects tend to undergo large degrees of nonrigid deformation throughout the sequence. Standard skeletal model-based motion capture techniques are not able to handle such motions.

Vision-based tracking techniques and new modeling techniques are beginning to tackle many of these issues. Much of our cartoon capture process builds on such vision based techniques.

Because we want to retarget the motion onto a new image, model or photograph, there are other unique challenges:

1. Much of the current retargeting techniques are based on skeletal models, and address challenges on how to map and scale parameters from one character's kinematics to a different character's kinematics. We circumvent those problems in using a key-shape based retargeting technique.

2. Since we capture from cartoons, we only have 2D information. Some of the investigated retargeting domains are 3D output models. This is, in general, an under-constrained setup. We also show how we can tackle 3D motion with 2D to 3D key-shape mappings.

3. In order to preserve the style, the absolute motion might differ significantly between input and output. For instance a small character with long arms will swing those arms differently than a bigger character with shorter arms. We can handle some of those mappings in defining different key-shapes and different interpolation functions for the input and output domain. But there are many other motion changes, due to different character physics, that we can not tackle. A "heavy walk" will remain a "heavy walk", no matter how the weight changes from input to output character.

1.4 Cartoon Capture and Retargeting

Figure 2 shows an overview chart of our technique. The input to cartoon capture is the digitized video, and a user-defined set of key-shapes (chosen from the source sequence). Cartoon capture transforms a digitized cartoon into a cartoon motion representation. We parameterize the motion with a combination of affine transformation and key-weight vectors. In this way, we can describe a wide range of motion and non-rigid shape deformations. For the cartoon retarget process, the user has to define for each input key-shape a corresponding output key-shape, or key-image, or 3D key-model. The motion parameters are mapped from the source to the target. We believe that by maintaining the timing and motion parameters from the original animation, we can maintain most of the "essence" of the expressive movement [Whitaker and Halas 1981].

The organization of this paper is as follows: after a discussion of related work in section 2, we introduce the cartoon motion representation in section 3.1. In section 3.2 and 3.3, we will describe two capture processes. Section 3.4 describes cartoon retargeting, using the examples we created. Finally, section 4 summarizes our results and discusses future work and improvement.

2 Related Work

Some previous work has addressed parts of the challenges above. [Gleicher 1998] describes a method for retargeting motions onto figures with different proportions. The retargeting problem was framed as a constraint optimization problem. [Popović and Witkin 1999] applied the principles of physics based animation and constraint optimization formulation on motion captured data. This method produces realistic motions and can be used on characters with different kinematic structure. Both of these methods require the input as joint angle data and applies the motion to articulated figures only.

Expression Cloning [yong Noh and Neumann 2001] introduces a method to retarget facial expression animation to another head model. This method maps the motion vectors of the source face mesh to the target mesh. This approach can be used for retargeting of non-rigid shapes, but it requires the input as a set of 3D motion vectors. It also assumes that the source and target object have very similar topology, otherwise the retargeting of motion vectors would not be meaningful. In cartoon retargeting, we want a variety of output formats, such as 3D models, 2D images and photographs, therefore a more flexible technique is necessary.

There are many other ways to parameterize motion in computer animation. Our work is most related to representing animation as interpolation of key shapes as is often used in facial expression animation [Parke 1972; Pighin et al. 1998]. Professional 3D animation tools also have software for creating animation from key-shapes. Most of this work focuses on the synthesis aspect of this method and are applied to the area of facial animation. Very little work addresses the inverse problem of extracting the weights for the key-shaped animation.

There has been extensive work on using Principle Component Analysis (PCA) and other sub-space techniques to estimate shape or motion variations [Bregler and Omohundro 1994; Lanitis et al. 1995; Black and Jepson 1996; Blanz and Vetter August 1999], or sequences of animation [Alexa and Müller 2000] but this has not been used for retargeting. So far, the focus of those papers is efficient data representation, learning, or compression applications. Furthermore, PCA decompositions result in a set of orthogonal basis vectors. While this is a mathematically elegant representation, the basis vectors do not have any physical meanings. In cartoon capture, we want to use key poses that visually make sense to the animators. The goal is that animators can design key poses for the output media of their choice by looking at the key poses for the input cartoon and retarget the motion onto the output characters. We also want something that is intuitive and easy to use. The animators can make sensible animations using this method without having to understand the details of the technology.

A related cartoon interpolation system has been demonstrated by [Ngo et al. 2000]. It uses a configuration-space model based on simplicial complexes, instead of a key-shape based model. It has only been demonstrated for interpolation tasks, but could also be extended to be used for the capture process.

3 Technical Approach

As mentioned previously, the goal of this project is to isolate the motion style of an existing cartoon animation and apply the same style to a new output domain. We first describe the parametric motion representation. Then, we explain how we can capture the parameters from cartoon input. Finally, we show how we apply the captured motion style to different output characters.

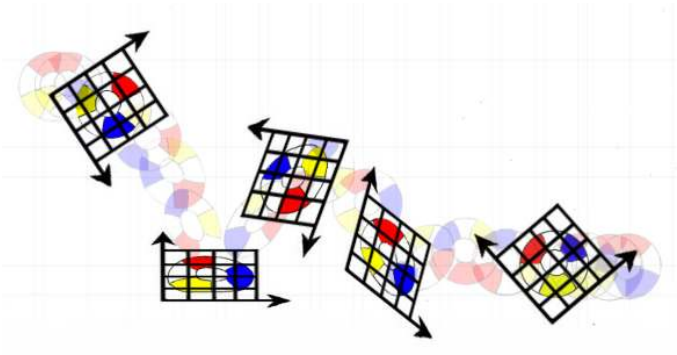

Figure 3: Shows the motion of the bouncing ball encoded in terms of the 6 affine parameters.

3.1 Modeling Cartoon Motion

We describe cartoon motion as a composition of two types of deformations: 1) Affine deformations, that encode the global translation, rotation, scaling, and sheer factors, and 2) Key-shape deformations, that are defined relative to a set of key-shapes.

3.1.1 Affine Deformations

An important part of cartoon motion comes from the velocity of the entire body, and how it stretches and squashes in different directions. We demonstrate this on the bouncing-ball motion in Figure 3. The motion style is determined by how fast the ball travels, the arcing (shape of the ball trajectory), how much the ball rotates from frame-to-frame, how much it squashes and stretches, and the timing of the squash and stretch. We can approximate the motion style with a sequence of affine deformations as illustrated with the overlayed grid in Figure 3.

The ball shape S is deformed to a shape $V(t)$ at time frame t with affine parameters $\theta(t) = [a_1, a_2, a_3, a_4, d_x, d_y]$:

$$V = warp(\theta, S) = \begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} \cdot S \qquad (1)$$

a_1, a_2, a_3, and a_4 describe rotation, x/y scale, and shear, and d_x, d_y code the x/y translation. S is a $3 \times N$ shape matrix $[s_1, ..., s_N]$ coding N points in homogenous form $s_i = [x_i, y_i, 1]^T$. For instance, S could be a set of points along the contours of the ball. If we replace S with a new contour, for example a donut-shape, or a photograph, but apply the same affine motions $\theta(1), ..., \theta(t)$, the moving shapes $V(1), ..., V(t)$ completely change, but the motion style remains the same (See video).

3.1.2 Key-Shape Deformations

Consider a more complicated motion such as the frog jumping on the left part of Figure 2. With affine parameters, we can approximate the coarse motion, but we miss several important deformations, such as the extension and contraction of the legs. To cover those deformations, we use a set of characteristic key-shapes S_i (or blend-shapes). These shapes are picked by the user, and should include all possible extreme deformations. Figure 2 shows three example key-shapes. The example shows how the frog transforms from a stretched-out shape S_1 in the air to a squashed shape S_2 in landing. All in-between shapes can be approximated as multi-way linear interpolations. Our motion model extends to:

$$V = warp(\theta, S_1 ... S_k) = \begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} \cdot (\sum_k w_k \cdot S_k) \qquad (2)$$

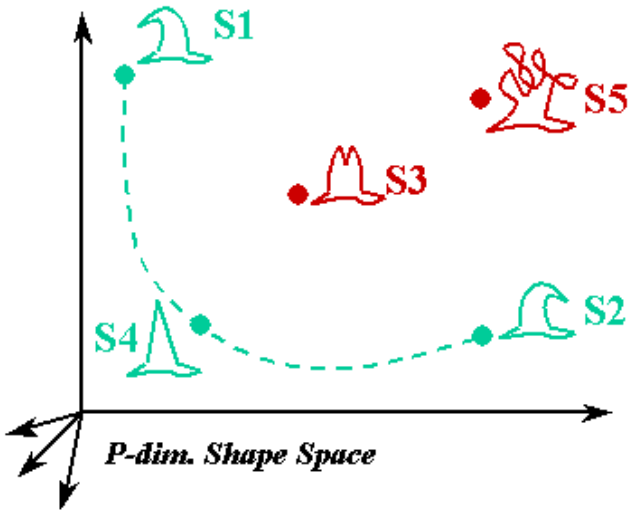

Figure 4: Multi-dimensional Warping Space

The cartoon shape V is now parameterized by $6+K$ variables, $\theta = [a_1, a_2, a_3, a_4, d_x, d_y, w_1, ..., w_k]$, the 6 affine parameters (as defined previously), and the K key-shape interpolation weights.

3.1.3 Extended linear warping space

In some domains linear interpolation is a good approximation, but in many domains, it produces in-between shapes with undesirable visual artifacts. Restrictions of linear interpolations are demonstrated in Figure 4. (Shape S_3 is the linear interpolation (average) of shape S_1 and S_2).

It is possible to approximate those in-between shapes with additional linear key-shapes. For example adding shape S_4 as a new key-shape allows us to better approximate the visually meaningful in-betweens of S_1 and S_2. In practice, a large number of such additional key-shapes are needed. We like to avoid this, since it puts additional burden on the user for defining the large amount of source and retarget key-shapes.

Another possibility is to replace the linear warping function with a nonlinear function. For instance, [Alexa et al. 2000] proposed a nonlinear warping function, that avoids many visual artifacts. Using such a function helps us in keeping the number of key-shapes to a compact size, but introduces two new challenges: 1) Alexa's and many other nonlinear functions are well defined for the interpolation between two key-shapes, but multi-way warping functions are non-trivial. 2) These functions are highly nonlinear (especially the multi-way extensions), and their inverse function (that is needed for cartoon capture) is numerically challenging due to many local minima and singularities.

We propose a technique that avoids those challenges: We use the Alexa function as a preprocessing step to enlarge the linear key-shape set: 1) We automatically generate M in-between shapes for each pair of hand-picked key-shapes S_i,S_j. This produces a large set of $(K-1)\times(K-2)\times M$ shape examples, where K is the number the original key shapes. It densely covers the entire cartoon shape space. 2) We apply Principal Components Analysis (PCA) on this large shape database [Jollife 1986]. PCA will generate a mean shape M, and eigen-vectors E_1, ..., E_L that span principal shape variations in orthogonal directions of the shape space. Every original example can be approximated with $V = M + \sum_l \alpha_l \cdot E_l$. The number of eigen-vectors is determined by the maximum allowed approximation error. If we constrain $\sum_l \alpha_l = 1$, we can write:

$$V = \alpha_{L+1} \cdot M + \sum_{l=1}^{L} \alpha_l (M + E_l) = \sum_{l=1}^{L+1} \alpha_l \cdot S_l \qquad (3)$$

Therefore the extended shape space is computed in setting $S_l := M + E_l$ and $S_{L+1} := M$. Usually the number of automatically derived key-shapes is higher than the original hand-picked shapes (to cover the nonlinear in-betweens), but sometimes it can be lower, when the user has chosen redundant key-shapes.

The *extended linear warping space* gives us the modeling power of more complex nonlinear warping functions, but we keep all numerical advantages, due to the linear key-shape representation. Furthermore, this technique allows us to transform a function (like Alexa's) that is only defined for shape pairs into a multiway warping function of more than 2 key-shapes. This technique will show its full potential especially for our video-capture process, as we will describe later.

3.1.4 Sub-Part Decomposition

Simple characters like the bouncing ball example or the frog deformations can be modeled with a global set of affine and key-shape parameterizations, but more complicated characters should be split into sub-parts. For example, in articulated figures, the leg deformation can be modeled separately from the arm deformations, and the head deformations.

In the following section we describe three tasks that use the motion model (2): 1) Contour capture, that assumes a known contour V and solves for θ, 2) Video capture, that applies (2) directly to the unlabeled video to solve for θ, and 3) Cartoon-retargeting, which takes a θ and different S_i to generate a new shape motion V.

3.2 Contour Capture

The input is a sequence of cartoon contours: $V(1), ..., V(t)$ and the hand-labeled key-shapes $S_1, ..., S_K$. If the animator used a computer tool, those V vectors can come directly from the tool, or the vectors will be hand-rotoscoped from stock-footage animation. Given V, the equation (2) is used to solve for θ.

Since our cartoon motion model will not exactly match the contour input, we estimate a motion vector θ that will approximate the contour input. This can be done in minimizing the following error term:

$$Err = ||V - warp(\theta, S_1, ..., S_k)||^2 \qquad (4)$$

We minimize this term with a 2 step procedure. 1) First we compute the affine motion parameters, 2) We estimate the key-weights on affine aligned shapes.

Affine Capture: Estimating affine motion of contours can be done with a closed-form solution. We measure the affine motion in minimizing the following error term:

$$Err_{aff} = ||V - \begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} \cdot S_1||^2 \qquad (5)$$

The standard least-squares solution is

$$\begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} := V \cdot S^T (S \cdot S^T)^{-1} \qquad (6)$$

Key-Weight Capture: Now we need to find the set of key-weights w_i in minimizing the full approximation error:

$$Err = ||V - \begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} \cdot \sum_k w_k \cdot S_k||^2 \quad (7)$$

We experienced improved results if we put additional constraints on the key-weights w_k. So far we have ignored the fact that the shape space of $S_1, ..., S_k$ is much larger than just the visually correct in-between shapes. We would like to avoid the case that noise and other small (unmodeled) variations of V will cause very large positive and negative key-weights w_i. This causes severe problems for the retargeting task. Many "illegal" shapes can be generated with large weights. Usually most in-between shapes are generated by using only a few key-shapes (only a few w_i are non-zero, and they sum to 1). Furthermore, the key-shapes work best for interpolation, but only have limited power for extrapolation.

We enforce this with the following constraints:

- Constraint 1: Only J interpolation weights are non-zero. This enforces that each possible shape lies in a smaller (local-linear) J dimensional subspace. Such shapes are closer to the blue curve in Figure 4.
- Constraint 2: All key-weights add to 1. If all key-shapes have the same volume, and we also constrain the affine matrix A to be a rotation matrix (without scale change), this constraint approximates an important animation principle called "preservation of volume". The volume preservation is violated, if the key-shapes are too far from each other (like S_1 and S_2 in figure 2).
- Constraint 3: All weights must lie in a margin [T1-T2]. Usually T1 = -0.5 and T2 = 1.5. This enforces limited extrapolation.

Minimizing the quadratic term of (7) due to the linear equality and inequality constraints can be done with quadratic programming [Gill et al. 1981]. We use an implementation that is available in the optimization toolbox in Matlab.

Since the affine estimation was done relative to S_1 and not to the weighted combination of all key-shapes, we need to iterate. Given the key-weights, we compute the weighted interpolation S. The new affine parameters are estimated based on S. We then re-compute the key-weights based on the new affine adjusted V, and iterate until convergence.

3.3 Video-Capture

As in contour-capture, the output of video-capture is a sequence of motion vectors θ that fit the input data. Now our input data is a sequence of images I instead of contours. We extend our cartoon motion model such that it directly models image pixel variations. With this extension we can incorporate the cartoon motion model into a vision-based region tracking technique [Lucas and Kanade 1981; Shi and Tomasi 1994; Bergen et al. 1992]. This allows us to track cartoons without contour labels.

We use the following notation: $S^{2\times N} = [s_1, ..., s_N]$ contains the x/y coordinates of all N pixels of the cartoon image region (Figure 5). $I(s_i)$ is the graylevel value of an image I at pixel location s_i. $I(S)$ denotes the vector of all pixel graylevels in the cartoon region. And $I(warp(S,\theta))$ is the warped image using the $warp(S,\theta)$ function. As in contour-capture, we cannot exactly match the image warp with our motion model (due to noise and inaccuracies of our model). We therefore want to minimize the following error function:

$$err_{image} = ||I_t(warp(\theta, S)) - I_0(S)||^2 \quad (8)$$

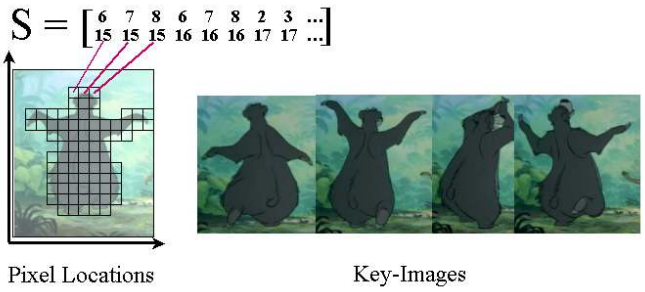

Figure 5: Left side illustrates how the shape vector S is used to index all pixel locations of the cartoon. Right side shows four key-shape examples. ©Disney.

This equation states that if we warp I_t it should look similar to I_0. The error term of (8) is in a sense the previous error term (7) mapped through the image function I.

Affine Capture: For affine motion only, we can rewrite equation (8) to:

$$err_{aff} = \sum_{s_i \in S} (I_t(A \cdot s_i + D) - I_0(s_i))^2 \quad (9)$$

where A is the rotation, scale and shear part of the affine parameters, and D is the translation. This error function can be minimized using the well known affine version of the Lucas-Kanade technique [Bergen et al. 1992; Shi and Tomasi 1994] (a standard least-squares based technique in the computer vision literature). It is beyond the scope of this paper to explain this technique in full detail, but we can briefly summarize the estimation:

We can not directly apply linear least-squares estimation, since $I_t(A \cdot s_i + D)$ is nonlinear. [Lucas and Kanade 1981; Horn and Schunk 1981] use following linear approximation of I_t:

$$I_t(A \cdot s_i + D) \approx I_t(s_i) + [I_x(s_i), I_y(s_i)] \cdot (A \cdot s_i + D) \quad (10)$$

where $\Delta I = [I_x, I_y]$ is the image gradient of I_t in x and y direction. We use a 2D Gaussian convolution filter as in [Canny 1986] to estimate the image gradient in a noise-robust way. Equation (9) can we rewritten to:

$$\begin{aligned} err_{aff} &\approx \sum_{s_i \in S} (I_t(s_i) + \Delta I(s_i) \cdot (A \cdot s_i + D) - I_0(s_i))^2 \\ &= \sum_{s_i \in S} (H_i \cdot \theta_{aff} + z_i)^2 \quad (11) \\ \theta_{aff} &= [a_1, a_2, a_3, a_4, d_x, d_y]^T \\ H_i &= [I_x(i) \cdot x_i, I_x(i) \cdot y_i, I_y(i) \cdot x_i, I_y(i) \cdot y_i, I_x(i), I_y(i)] \\ z_i &= I_t(i) - I_0(i) \end{aligned}$$

The standard least-squares solution of this linearized term is

$$\theta_{aff} = (H^T \cdot H)^{-1} \cdot H^T \cdot Z \quad (12)$$

with

$$H = \begin{bmatrix} H_1 \\ ... \\ H_N \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} z_1 \\ ... \\ z_N \end{bmatrix} \quad (13)$$

Since we use the linear approximation (10), the optimal motion parameter θ is found in using equation (11) iteratively in a Newton-Raphson style minimization.

Affine and Key-Shape Capture: If our motion model includes the affine and key-shape deformation model, we need to further extend the estimation framework. We replace the image template I_0 with a combination of L key-images: $\sum_l w_l E_l$:

$$err_{keys} = \sum_{s_i \in S} (H_i \cdot \theta_{aff} + I_t(s_i) - \sum_l w_l \cdot E_k(s_i))^2 \quad (14)$$

$$= \sum_{s_i \in S} ([H_i, E_1(i), ..., E_L(i)] \cdot \theta + I_t(i))^2 \quad (15)$$

The extended vector $\theta = [a_1, a_2, a_3, a_4, d_x, d_y, w_1, ..., w_L]^T$ can be estimated with standard least squares estimation as above.

The right side of Figure 5 shows example key-images for the Baloo sequence. Those are the original hand picked key-shapes. Since the linear interpolation of those hand picked key-images produce "illegal" shapes (linear interpolating the arm motion merely generates a double image of both arm configurations), it is essential, that we use an extended key-shape set as described in 3.2. Many in-between images are generated from the key-shapes using the non-linear Alexa-warping function. Applying PCA to the enlarged dataset, resulted in L "eigen-images" E_l. This estimation framework is closely related to [Black and Jepson 1996; Lanitis et al. 1995; Blanz and Vetter August 1999].

We had good experience with generating 5 inbetween images for each hand-picked key-image. For the tracking we used $L = 10$ eigen-images, and the algorithm converged usually after 40 iterations. We did not quantify the accuracy of the tracking, but show for all sequences the tracking points in the video.

Sub-Part Layers: The video-capture process is very sensitive to outliers. Outliers are pixels that are not part of the cartoon region. They could be part of the background, or occluding foreground (including self-occlusion from other cartoon parts). We discount those pixels automatically in computing an adaptive alpha matte. Usually, the cartoon region has a specific color range. We can generate an alpha-matte in using a probabilistic color segmentation technique. For instance, the blue hat in Figure 3.3 can be segmented automatically. The second row in Figure 3.3 and Figure 7 shows some examples. Our video-capture process is then only performed on pixels that are included in this matte. This can be done by reducing the summation to only those pixels:

$$err_{keys} = \sum_{s_i \in Layer} (I_t(A \cdot s_i + D) - \sum_l w_l \cdot E_k(s_i))^2 \quad (16)$$

3.4 Retargeting Cartoon Motion

We experiment with different output media, including 2D cartoon animation, 3D CG models, and photo-realistic output. For each domain, we need to model how a specific input key-shape looks in the output domain. For simple affine deformations, we simply replace the input template with a template in the output domain. For key-shape models, we need to design the corresponding key-shape and the interpolation function in the output domain. The corresponding output shape can look similar or drastically different from the input as long as the key poses are consistent in their meanings. For example, if key pose 1 in the input cartoon is more extreme than key pose 2, then key pose 1 in the output image should also be more extreme, in the same way, than key pose 2 in the output image.

3.4.1 Designing the Output Model and Retargeting

2D Drawing and Photographs: For each key-shape used in the key-shape deformation, a corresponding 2D shape is drawn in the output domain. In addition, the corresponding control points

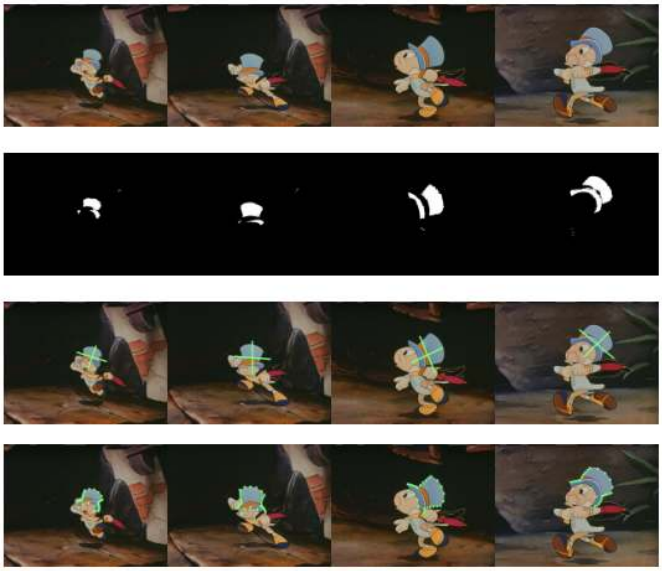

Figure 6: The second row shows the hat color segmentation, the third row shows the affine capture, and the fourth row shows the key-shape based tracking of the hat. ©Disney

Figure 7: Shows the color-clustered layer and the key-shape based tracking of Baloo. ©Disney

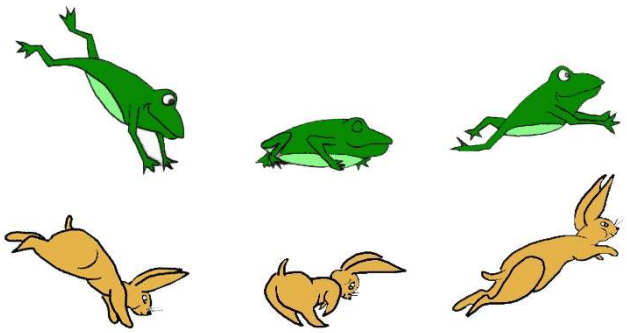

Figure 8: 2D example of key-shapes for the input cartoon and corresponding output key-shapes

(or contours) between the different output key-shapes are labeled. Figure 8 shows the key-shapes of a frog and the corresponding key-shapes of a bunny. The retargeting process is as follows: First, we extract the affine motion of each output key-shapes with respect to some reference frame using equation (5). Then, we apply the chosen interpolation function to the affine adjusted key-shapes using the weights obtained from the cartoon capture process. Finally, the affine motions of the input cartoon are added to the resulting key-shapes to produce the final animation.

3D Models: To retarget the input motion to 3D models, the animator uses a 3D modeling tool to make the key-shapes for the output animation. To retarget the affine motion of the input cartoon to a 3D model, we need the equivalent of affine motion in 3D. For the in-plane motion (image plane), we map the affine parameters just as in the 2D case. We do not explicitly recover the out-of-plane motion from the input cartoon. This does not however imply that the models are flat. The 3D information is inferred when we design the key poses for the output 3D model. For example, as shown in the top row of Figure 9, it is difficult to measure which direction the character's right foot is pointing from the input cartoon drawing. The animator, while doing the 3D modeling, interprets the right foot as pointing toward the viewer in the first key-shape, as shown in the bottom row of figure 9. Since the solution to the equation is such that the weight has to be one for that key-shape and zero for the other keys, the retargeted output character naturally points his right foot toward the viewer.

Applying the key-shape deformation to 3D models works the same way as the 2D examples, except now we might use control vertices for the mesh or the nurb, depending on the modeling choice. Linear interpolation is used for all the 3D examples. However, we make a special case for retargeting to 3D articulated figures. We choose to interpolate in the joint angle space since interpolating in the joint-angle space is analogous to the non-linear shape interpolation function described in section 3.1. The 3D joint angles are first taken from the 3D model. The joint angles for the in-between frames are then interpolated using the weights from the capture process. Note that this is different from interpolating each joint angle independently. All the joint angles are interpolated with the same weights from the cartoon capture process and more than 2 key-shapes can be used for each output pose.

Additional Constraints and Post-processing: In many cases, the translation of the original cartoon needs to be modified to satisfy certain constraints in the output domain. This is due to the fact that we factor out the affine parameters of the output key poses and apply the affine parameters of the input cartoon to the output sequence. Since the output character can have different proportion

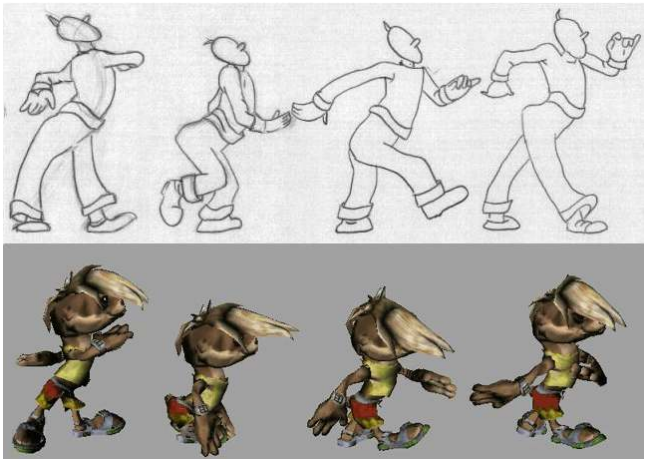

Figure 9: 3D example of key-shapes for the input cartoon and corresponding output key-shapes

Figure 10: The hat retargeting sequence. The hat is replaced with a witch hat. ©Disney

and dimensionality from the input character, using the same affine motion results in undesirable effects, such as the output character's foot going through the ground. In most cases, we found that simple ad-hoc global translations produced a reasonable result. These global translations include constraints that ensure that the foot is at a certain position at a specific time frame.

4 Examples

To demonstrate and test cartoon capture, we create several examples. First, we use the video-capture technique described in section 3.3 to follow the motion of an animated hat that we then retarget onto a simple two-dimensional drawing of a different hat. We refit the new hat onto the original footage. Figure 10 shows few frames and the video shows the entire capture and retargeting experiment.

We then capture the dance of Baloo from *The Jungle Book* and retarget the motion to a drawing of a flower. Again, we use video-capture techniques to extract the motion parameters and apply them on the output drawing of the flower. Figure 11 shows the results.

We capture the broom motion from the Sorcerer's Apprentice sequence of *Fantasia*, and retarget the motion onto a digitized photo of a broom. We use contour-capture with constraints and retargeting with additional constraints as described in section 3.4, since we change the broom dimensions. Then we composite the broom sequence onto a live-action video sequence. Figure 12 shows several retargeted brooms.

We also experiment with capturing only one contour, the line-of-action, as the source of the motion. The line-of-action is a very important principle in tradition animation. The line-of-action is a

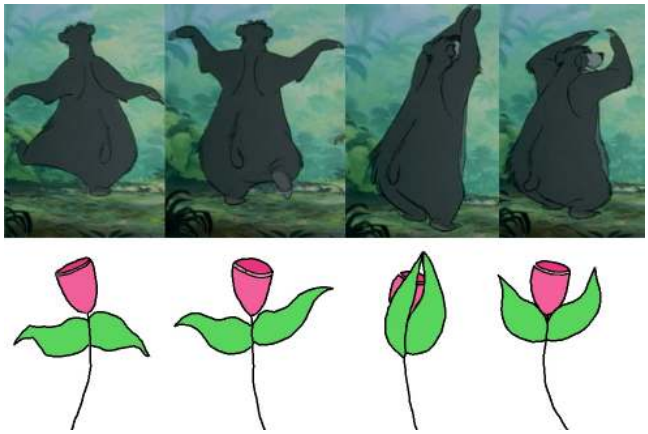

Figure 11: Tracking of Baloo's dance and retargeting to a flower. ©Disney

Figure 12: Broom Retargeting Sequence. (The source is not shown here – It is the famous sequence from the Sorcerer's Apprentice in Disney's *Fantasia*)

single line running through the character, that represents the overall force and direction of each drawing. Generally, before drawing the full character, an animator draws in the line-of-action to help determine the position of the character. By simply changing the line-of-action – making it more curved, sloped or arched in a different direction – the entire essence of the drawing can be changed. We experiment with using the line-of-action as the source in order to understand how much information we can get from a single contour line and as a starting point for future work using minimal information to achieve the greatest results. In this example, we track the line-of-action from a classic Porky Pig sequence and retarget the motion of the single contour onto a 2D character. Although there is not enough information in this contour to solve for more complex motion, such as how the legs move relative to each other, the essence of the motion is still present in the retargeted output. Figure 13 illustrates this example.

We also create examples of cartoon captured motion, retargeted onto 3D models. In the first example, we capture the motion from a walking character and retarget to a 3D model of an otter. In Figure

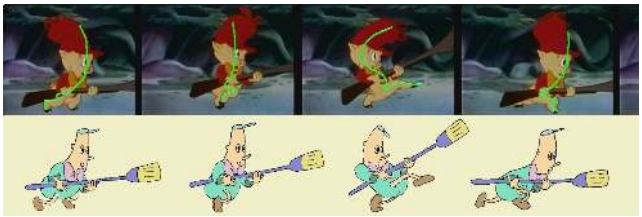

Figure 13: Capturing line-of-action and retargeting of Porky Pig images from Corny Concerto (1943), onto a new 2D character

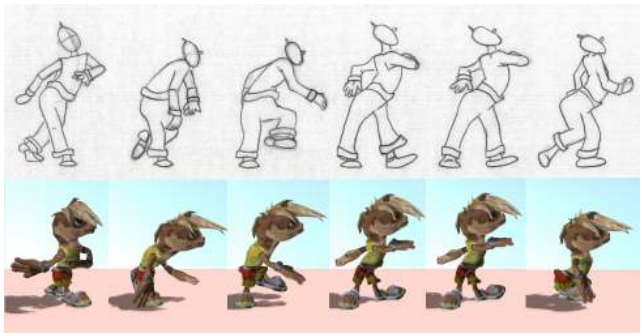

Figure 14: Motion of walking cartoon character retargeted to 3D model

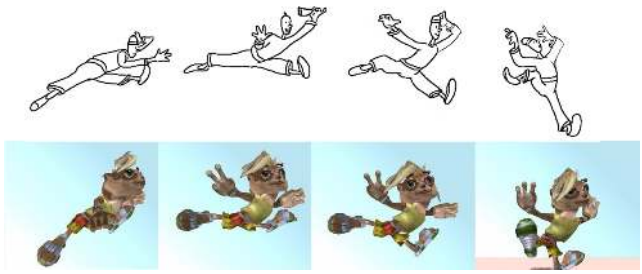

Figure 15: Motion of jumping character retargeted to 3D model

9, we show some of the key poses.

Figure 14 shows some final rendered frames in the retargeted sequence. Again, please consult the video for more information.

In the next example, we capture a cartoon character jumping in a way that is impossible for an average human, and retarget onto the same 3D model. Again, contour capture is used to estimate the motion parameters. Figure 15 shows some frames from the retargeted sequence.

The video shows all results for the different target media. Our experiments in cartoon capture provide an opportunity to test the cartoon capture concept and technique. While the results have room for technical improvement, they clearly demonstrate that the concept has merit. The motion style from the captured cartoon is successfully translated to the new image. The timing is preserved and the key-shapes from the original animation are mapped onto the new animation. The animations with the cartoon capture process are expressive and compelling. The personalities of the original cartoon characters are transferred to the target images.

5 Discussion

We have demonstrated a new technique that can capture the motion style of cartoons and retarget the same style to a different domain. This is done by tracking affine motion and key-shape based interpolation weights. We described two different versions of capture, one starting with cartoon contours, and one that can be applied directly to unlabeled video. Both techniques are based on least-squares techniques similar to other estimation techniques commonly used in computer vision. The new contribution of our technique is the use of an extended linear warping space and additional constraints on the interpolation weights. The extended PCA space allows us to approximate any nonlinear warping function, without increasing the complexity or lowering numerical robustness for the video capture. Furthermore, the linearized PCA space allows us to perform multiway warping with more than two key-shapes, even if the non-

linear function we approximate, does not offer this feature. The new constraints on the interpolation weights are usually not necessary for tracking applications only, but has proven useful for retargetting domains.

With our strategy of designing input-output key-shape pairs, we circumvented many problems that arise in standard skeleton based motion adaption.

We believe this work is significant for its technical contribution, but more importantly, this project attempts to bridge the gap between techniques that target the traditional expressive animation world and the motion capture based animation world. Certain domains require increased realism, and motion capture data is a more appropriate source, but many other domains call for more expressive and stylistic motion, and traditional animation data is the appropriate source. We have shown, that we can treat traditional animation footage in ways similar to what has been done with motion capture data. Therefore we hope to create new opportunities for researching and using a wider range of motion styles.

The results, while compelling for many reasons, have room for improvement. We addressed several of the challenges listed in section 1, but only touched on many other challenges. The 2D to 3D mapping is possible, but puts a large burden on the key-shape designer. The more complex the motion of a character is, the more key-shapes are needed. In some very complex domains, it might even be more labor to create the key-shapes, than to animate in the new domain directly by hand. Furthermore, the capture techniques can deal with very complex and challenging motion, but also need to be improved in accuracy. We have not employed any temporal constraints, like smoothness or specific dynamic model constraints. Many of our animations contain jitter, although very often the jitter is not that distracting, since the exaggerated motion dominates the perception.

Some of our future research directions will include how to incorporate further dynamical constraints to increase the robustness. Also, we plan to experiment with alternative minimization techniques for the capture process. Sampling techniques have been proven to be more robust for video motion tracking.

Currently, we are investigating methods to allow editing of motion styles. This would allow a single source motion to be used in a variety of scenes. We also want to work on filters for motion capture that add animation principles to create more expressiveness in the motion and allow a single motion capture sequence to have many motion styles.

Acknowledgements

We would like to thank Pat Hanrahan, Marc Levoy, James Davis, Henrik Wann Jensen, Maneesh Argrawala, Li-Yi Wei, Charles Ying, Kingsley Willis, Rahul Gupta, Erica Robles, Leslie Ikemoto, Alejandra Meza, Alana Chan, Jessica Bernstein-Wax, John Gerth, Ada Glucksman, Heather Genter, the Stanford Graphics Lab, and the SIGGRAPH reviewers for their significant help, support, and input on this project. We would like to thank Catherine Margerin, Greg LaSalle, and Jennifer Balducci from Rearden Steel for their invaluable help with the motion capture and retargeting tasks. We would like to thank Craig Slagel, Steve Taylor, and Steve Anderson from Electronic Arts for providing and helping us with the 3D Otter model in Maya. We would like to thank Disney Feature Animation, and especially Lance Williams, Stephanie Carr, and Bob Bacon for giving us the permission to use and reprint some of the famous Disney Animation Cells.

This research has been funded by the National Science Foundation, Electronic Arts, Microsoft Corporation, Sony, Intel, and Vulcan Ventures through the Stanford IMTV project, and the Stanford OTL office.

References

ALEXA, M., AND MÜLLER, W. 2000. Representing animations by principal components. 411–418. ISSN 1067-7055.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Computer Graphics Proceedings, Annual Conference Series, 157–164. ISBN 1-58113-208-5.

BERGEN, J., ANANDAN, P., HANNA, K., AND HINGORANI, R. 1992. Hierarchical model-based motion estimation. In *ECCV*, 237–252.

BLACK, M., AND JEPSON, A., 1996. Eigentracking: robust matching and tracking of articulated objects using a view-based representation.

BLANZ, V., AND VETTER, T. August 1999. A morphable model for the synthesis of 3d faces. *Proceedings of SIGGRAPH 99*, 187–194. ISBN 0-20148-560-5. Held in Los Angeles, California.

BREGLER, C., AND OMOHUNDRO, S. 1994. Surface learning with applications to lipreading. In *Advances in Neural Information Processing Systems*, Morgan Kaufman, San Francisco, A. Cowan, Tesauro, Ed.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell. 8*, 679–698.

GILL, P., MURRAY, W., AND WRIGHT, M. 1981. *Practical Optimization*. Academic Press, London, UK.

GLEICHER, M. 1998. Retargeting motion to new characters. In *SIGGRAPH 98 Conference Proceedings*, Addison Wesley, M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, 33–42. ISBN 0-89791-999-8.

HORN, B., AND SCHUNK, G. 1981. Determining optical flow. *Artificial Intelligence 17*.

JOLLIFE, I. T. 1986. *Principal Components Analysis*. New York: Springer.

LANITIS, A., TAYLOR, C., COOTES, T., AND AHMED, T. 1995. Automatic interpretation of human faces and hand gestures using flexible models. In *International Workshop on Automatic Face- and Gesture-Recognition*.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. *Proc. 7th Int. Joinnt Conf. on Art. Intell.*.

NGO, T., DANA, J., CUTRELL, D., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *SIGGRAPH 2000 Proceedings*.

PARKE, F. 1972. Computer generated animation of faces. In *ACM National Conferences*, 451–457.

PIGHIN, F., HECKER, J., LISCHINSKI, D., SZELISKI, R., AND SALESIN, D. H. 1998. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH 98 Conference Proceedings*, Addison Wesley, M. Cohen, Ed., Annual Conference Series, ACM SIGGRAPH, 75–84. ISBN 0-89791-999-8.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *SIGGRAPH 99 Conference Proceedings*, ACM SIGGRAPH.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *CVPR*.

WHITAKER, H., AND HALAS, J. 1981. *Timing for Animation*. FOCAL PRESS LTD.

YONG NOH, J., AND NEUMANN, U. 2001. Expression cloning. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 277–288. ISBN 1-58113-292-1.