Character Animation from 2D Pictures and 3D Motion Data

ALEXANDER HORNUNG, ELLEN DEKKERS, and LEIF KOBBELT
RWTH-Aachen University

This article presents a new method to animate photos of 2D characters using 3D motion capture data. Given a single image of a person or essentially human-like subject, our method transfers the motion of a 3D skeleton onto the subject's 2D shape in image space, generating the impression of a realistic movement. We present robust solutions to reconstruct a projective camera model and a 3D model pose which matches best to the given 2D image. Depending on the reconstructed view, a 2D shape template is selected which enables the proper handling of occlusions. After fitting the template to the character in the input image, it is deformed as-rigid-as-possible by taking the projected 3D motion data into account. Unlike previous work, our method thereby correctly handles projective shape distortion. It works for images from arbitrary views and requires only a small amount of user interaction. We present animations of a diverse set of human (and nonhuman) characters with different types of motions, such as walking, jumping, or dancing.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation;
I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms: Algorithms

Additional Key Words and Phrases: 2D character animation, 3D motion data, camera and model pose determination, as-rigid-as-possible shape manipulation with perspective correction

ACM Reference Format:

Hornung, A., Dekkers, E., and Kobbelt, L. 2007. Character animation from 2D pictures and 3D motion data. *ACM Trans. Graph. 26*, 1, Article 1 (January 2007), 9 pages. DOI = 10.1145/1186644.1186645 http://doi.acm.org/10.1145/1186644.1186645

1. INTRODUCTION

In recent years, research on 2D image manipulation has received a huge amount of interest. Very powerful solutions for problems such as matting [Sun et al. 2004], image completion [Drori et al. 2003], texture synthesis [Efros and Leung 1999], and rigid image manipulation [Igarashi et al. 2005] have been presented. Based on these and similar methods, it has now become possible to explore interesting new ideas to reanimate still pictures, for example, as done by Chuang et al. [2005] in their article on animating pictures using stochastic motion textures. They animate passive elements, such as water and trees, that are subject to natural forces like wind. In this article, we want to take the idea of creating animations directly in image space one step further by making photographed persons move.

One possible approach to address this problem would be the reconstruction of a textured 3D model, and to animate this model using classical animation techniques. This, however, would require complex, fully textured 3D models which have to be created and adapted per image. In particular, for highly detailed characters such as the Scarecrow example shown in Figures 2 and 7, the required manual model-adaption process would be impractical. Moreover, it would be necessary to apply very sophisticated 3D rendering techniques to realistically embed the 3D model into the 2D image so as to preserve the photorealism or style of the original input image. By contrast, even simple 2D morphing and blending often leads to more convincing results than using sophisticated 3D reconstruction and rendering. For instance, methods such as Poisson-matting or 2D image completion allow for a smooth and realistic combination of different image contents, which is much harder to achieve when trying to augment 2D images with 3D models.

Hence, we present a purely image-based approach to combine realistic images with realistic 3D motion data in order to generate visually convincing animations of 2D characters. This is motivated by the intention to preserve the realism or style of the original image data, without losing quality due to intermediate conversion steps into a 3D representation. Several recent articles [Barrett and Cheney 2002; Igarashi et al. 2005; Chuang et al. 2005] have shown the promising potential of such solely image-based animation approaches.

The contribution of this article is a method to generate animations of photographed or painted 2D characters based on 3D motion data. For arbitrary input images, our method robustly reconstructs the camera and 3D model pose corresponding to the depicted subject. Using generic shape templates, the character is decomposed into animation layers. Occluded regions and the background are reconstructed by texture synthesis. We show how the resulting character-shape can be animated using an augmented version of the

The CMU motion database used in this work was created with funding from NSF EIA-0196217.

Authors' address: A. Hornung, E. Dekkers, and L. Kobbelt, Computer Graphics Group, RWTH-Aachen University, Lehrstuhl für Informatik 8, RWTH-Aachen, Ahornstraße 55, 52074 Aachen, Germany; email: {hornung, dekkers, kobbelt}@cs.rwth-aachen.de.

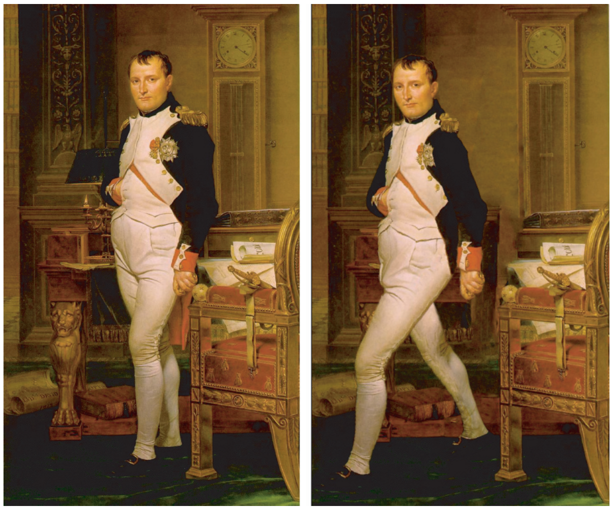

Fig. 1. Given only a single input image (left), our method generates 2D animations from 3D motion data (right).

as-rigid-as-possible shape manipulation technique [Igarashi et al. 2005], which correctly handles projective distortion effects such as foreshortening. In combination, these techniques enable us to realistically change the pose of a character or create animations from single input images of arbitrary human and nonhuman subjects based on 3D motion data.

2. RELATED WORK

Our work is inspired by a diverse set of interesting methods for augmenting and animating 2D images. When manipulating images, inevitable prerequisites are tools to generate proper segmentations or alpha mattes [Sun et al. 2004] and the completion of missing image information [Drori et al. 2003; Sun et al. 2005; Pavić et al. 2006].

These techniques have paved the way for different kinds of image manipulation approaches. Corrêa et al. [1998] present a warping technique to augment cel animations with texture. A complete image-based modeling and editing system has been presented by Oh et al. [2001], which contains, among other things, custom-tailored algorithms to reconstruct approximate camera positions for different types of images. Bregler et al. [2002] capture the motion and style of cartoons and retarget this onto 2D images and 3D models. Barrett and Cheney [2002] present algorithms for manually deforming objects in 2D space by affine, warping-, or curve-based editing. Thorne et al. [2004] implemented a curve-based system which allows the user to animate characters by drawing motion lines. [Liu et al. 2005] decompose video sequences into different motion layers and show how the subtle motion of these layers can be emphasized. Chuang et al. [2005] present a method to animate passive objects in a picture (such as trees or water, for instance), which are subject to stochastically modeled natural forces. An object-space morphing technique for as-rigid-as-possible shape interpolation was first introduced by Alexa et al. [2000]. Igarashi et al. [2005] reformulate this problem and present an interactive method for rigid shape deformation in 2D. They simulate as-rigid-as-possible 2D shapes which can be arbitrarily deformed by the user in real time. This is achieved by imposing shape-preserving energy functionals on the interior of the shape. Our work is based on this method and extends it to the context of deforming multilayered shapes with perspective correction in order to simulate 3D deformations.

Further related to our work, Rademacher [1999] shows that view-dependent appearance changes in 3D rendered cel animations can be effectively simulated using a set of interpolated geometries. Ono et al. [2004] generate 3D character models from user-specified strokes, which allow them to add illumination and perspective texturing effects to 2D cel animation. In this work we use similar ideas by providing a set of “inflated” 2D shape templates used for generic character animation directly in image space.

The second class of important results in the context of our work can be found in the analysis and synthesis of human poses and motion. In 3D, motion-based deformation and animation of avatars has, for example, been discussed by Lewis et al. [2000] and Lee et al. [2002]. Different methods for motion analysis and video-based motion synthesis have been described by Jojic and Frey [2001] and Mori et al. [2004]. However, our work is more closely related to single pose analysis than the analysis of motion sequences, since our aim is the transfer of 3D human motion into a still image of a 2D shape, instead of video sequences.

Two problems we are faced with in our work are the unknown subject pose on the one hand, and the unknown camera model, on the other. Several publications have been presented to recover the pose of a human 3D model from an image. Taylor [2000] presents a solution to reconstruct articulated objects from point correspondences for orthographic camera models, which, however, leads to ambiguous configurations. Parameswaran and Chellappa [2004] solve the same problem, accounting for projective foreshortening effects of a simplified skeleton model. Different methods for recovering 3D human body poses from single images or video are discussed in Gavrila [1999] and Moeslund and Granum [2001]. While existing automatic methods for pose estimation, such as Agarwal and Triggs [2006], would possibly reduce the amount of user interaction, they generally require, for example, segmented input images or other prior knowledge about the depicted subject. But more importantly, such methods could also restrict the user’s degrees of freedom for creating animations, as we will discuss in Section 4.

The problem of estimating a camera position from 2D to 3D correspondences is one of the central problems in computer vision. The two resources most important for our work are Hartley and Zisserman [2003] and Triggs et al. [2000], which we will discuss in detail in Section 5.

3. OVERVIEW

In this section we describe the prerequisites of our method and provide a high-level description of the algorithm.

As mentioned in the introduction, the central idea of our method is to deform the 2D shape of an arbitrary character directly in image space using 3D motion data (publicly available from, e.g., CMU Graphics Lab Motion Capture Database [2007]). The 3D motion data consists of a sequence of poses of a predefined human skeleton model. Each pose contains the respective positions and orientations of the skeleton bones and joints in 3D space.

The first step maps the 3D data to the 2D image by reconstructing a proper camera model and a model pose that best fits the 2D character in the image. We achieve this by letting the user manually specify correspondences between 2D and 3D, simply by selecting the joint positions in the 2D image that correspond to the joints of the given 3D human model. This *joint selection* step is explained in more detail in Section 4.

Based on these 2D to 3D correspondences, we automatically compute a camera projection matrix and a best-matching pose from the 3D motion data repository, which provides the closest possible fit

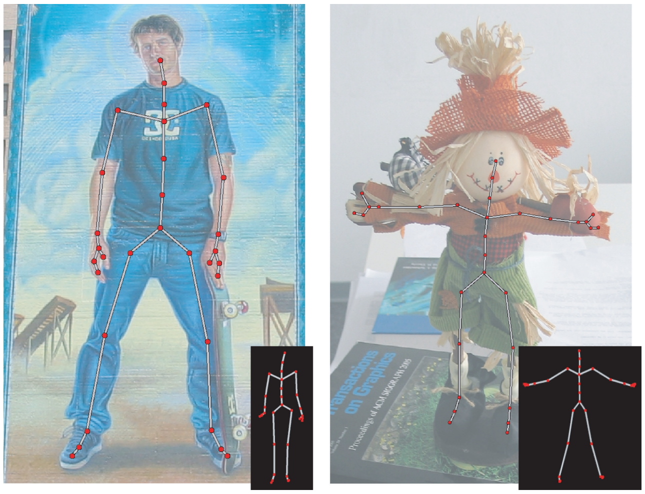

Fig. 2. The user-selected pose for the Skater and the Scarecrow example. The user simply drags the joint positions of a predefined human skeleton structure to their approximate 2D positions in the input image. Our method then (see Section 5) reconstructs an approximate camera position and a best-fitting model pose from the 3D motion data (overlayed images).

to the user-selected pose in 2D. This *camera and model pose determination* step is described in Section 5.

The next phase prepares the subsequent animation by an *initial model fitting* of a generalized shape template to the user-selected pose of the character. Each template consists of a set of layers corresponding to different parts of the character's body, combined into a single nonmanifold triangular mesh. To account for occlusion, textures and alpha mattes are pulled off the input image for each layer, using Poisson-matting. Missing parts of the background are synthesized by standard image completion (Section 6).

The final *animation* (Section 7) is rendered in real time using projected 3D joint positions from different poses in the motion sequence to control deformation of the 2D shape in image space. We demonstrate that the original as-rigid-as-possible (ARAP) shape manipulation is not sufficient for creating proper character animations, since it does not account for projective distortions, such as foreshortening, which naturally occur when projecting 3D motion data. We therefore propose an augmentation called *as-similar-as-possible* (ASAP) shape deformation, which properly takes these perspective effects into account.

This overall approach allows us to transfer inherently realistic 3D motion-captured data to 2D shapes of arbitrary human-like characters, while preserving the photorealism of the original image. The following sections explain each of these steps in detail.

4. JOINT SELECTION

To establish the necessary correspondences between the subject's 2D shape and the 3D skeleton model, we let the user manually select joint positions in the input image.

This is done by a simple interface, where the user can move the joints of a stylized skeleton having a structure compatible to our 3D data (see Figure 2). We preferred a manual user interaction over automatic procedures because this step generally takes just a few minutes to complete, and leads to superior results in poses which are ambiguous for automatic human-pose estimators, or which are difficult to estimate due to occlusions, for example. Furthermore, such methods would require a diverse dataset of example poses to work for arbitrary images. Hence, our manual method gives us more degrees of freedom to animate uncommon poses or perspectively deformed subjects, as in photographed paintings, and even nonhuman characters (see Figure 7).

Subsequently, the user chooses the type of motion which should be performed by the subject. Although it would be possible to estimate a "most likely movement" from the selected pose alone using the procedure explained in the next section, we believe that this additional user input is necessary to allow the user a controlled creation of a specific animation.

5. CAMERA AND MODEL POSE DETERMINATION

Since we want to use 3D motion data to animate the 2D shape, we have to compute a camera projection model P which describes the mapping from 3D joint positions of the skeleton to 2D image space. Furthermore, we need to find an optimal 3D pose $\mathbb{X}_o$ in our motion sequence that is closest to the manually selected 2D pose, so that we have a reasonable starting solution for the subsequent animation.

For the computation of the projection P, suppose we have a 3D pose given by a vector of joint positions $\mathbb{X} = (\ldots, \mathbf{X}_i^T, \ldots)^T$ which exactly correspond to the user-selected 2D joints $\mathbf{x}_i$. The unknown camera projection P can then be estimated from these world-to-image correspondences. A variety of linear and nonlinear algorithms exist for this problem [Hartley and Zisserman 2003; Triggs et al. 2000], which estimate P by minimizing, for example, the reprojection error from 3D to 2D: $\sum_{i,j} \|\mathsf{P}_j \mathbf{X}_i / \mathsf{P}_3 \mathbf{X}_i - \mathbf{x}_{i,j}\|^2$, with $j \in \{1, 2\}$ referring to the jth row of P and $\mathbf{x}_i$, respectively. However, since these methods often do not impose any constraints on P, the resulting projection generally does not correspond to a geometrically plausible Euclidean camera model. While such an unconstrained projective camera provides an optimal fit for the corresponding pose $\mathbb{X}$, it might lead to very unnatural deformations of the projected 3D model during the actual animation with different poses.

Instead, we would like to compute a parameterized camera projection $\mathsf{P} = \mathsf{KR}[\mathsf{Id}| - \mathbf{C}]$ consisting of an intrinsic calibration K, an extrinsic righthanded orientation R, and a world-space camera center $\mathbf{C}$. To impose the nonlinear orthonormality constraints on the extrinsic orientation, R has to be parameterized, for example, using Euler angles or quaternions. This yields a projection matrix consisting of 11 unknowns:

$$\mathsf{P} = \begin{pmatrix} \delta_x & s & p_x \\ & \delta_y & p_y \\ & & 1 \end{pmatrix} \mathsf{R}(\alpha, \beta, \gamma) \left[\begin{array}{c|c} \mathsf{Id} & -\begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} \end{array} \right] . \quad (1)$$

When computing a camera projection matrix using the preceding parameterization, a remaining problem, however, is the fact that an unconstrained optimization of the intrinsic calibration K still leads to undesired distortion effects during animation. The reasons for this are that we generally have to compute an initial (unconstrained) linear starting solution of P, and that it is unclear how to impose any meaningful constraints on K during the subsequent nonlinear optimization. This problem is complicated by the fact that the user-selected joint positions are generally relatively inaccurate, and because we are not guaranteed to have a perfectly matching pose in our 3D database in the first place. Hence, we have to constrain the optimization process to robustly converge to a reasonable estimate of P, despite the mismatch between the user input and the 3D motion data.

We found that, in the context of this work, a very stable method to estimate a camera P with the desired properties can be achieved by fixing the elements of K, except for the focal length during the optimization, and by providing a proper starting solution for the external data, instead of an unconstrained linear solution. Reasonable assumptions for the intrinsic data of most cameras are a zero-skew factor s, the principal point (p_x, p_y) at the image center and scaling factors δ_x and δ_y based on the image resolution, a unit aspect ratio, and a typical field-of-view (e.g., similar to an OpenGL projection matrix). Hence we set:

$$s = 0, \quad p_x = \frac{\mathsf{width}-1}{2}, \quad p_y = \frac{\mathsf{height}-1}{2}, \quad \delta_x = -\delta_y = \triangle f \frac{\mathsf{width}}{2\tan(\phi/2)}, \text{ with } \phi = \pi/8 \text{ and } \triangle f = 1 \ . \tag{2}$$

The factor $\triangle f$ allows us to adjust the predefined field-of-view ϕ during the nonlinear optimization process. The extrinsic data is simply initialized with $\mathsf{R} = \mathsf{Id}$ and a camera center at safe distance from the root of the 3D skeleton model, for example, a multiple of the bounding box diagonal d of the model:

$$\alpha = \beta = \gamma = 0, \quad \mathbf{C} = (0, 0, 5d)^T \ . \tag{3}$$

To compute an optimized projection matrix $\mathsf{P}(\triangle f, \alpha, \beta, \gamma, \mathbf{C})$ based on the remaining seven free parameters, we use an iterative Levenberg-Marquardt solver to minimize the reprojection error $\sum_{i,j} \|\mathsf{P}_j \mathbf{X}_i / \mathsf{P}_3 \mathbf{X}_i - \mathbf{x}_{i,j}\|^2$ with the aforementioned starting solution. Since this initial solution corresponds to a geometrically plausible camera model, and since the optimization process also works solely on the preceding parameterization of P, the resulting camera is guaranteed to preserve all desired intrinsic and extrinsic constraints. In all our experiments, this approach converged robustly to the desired optimum, without the necessity for any parameter adjustments. Moreover, it showed to be insensitive to parameter changes, that is, ϕ or $\mathbf{C}$.

To find the optimal 3D pose $\mathbb{X}_o$ for the current user selection, we couple the camera and model pose estimation into a single optimization problem with eight degrees of freedom by simply running the earlier algorithm over all poses contained in the motion data sequence, and set the pose resulting in the minimal reprojection error as $\mathbb{X}_o$ (see Figure 2). In contrast to other pose estimation techniques, we can drastically reduce the search domain for valid poses to one degree of freedom, since we only want to find the best solution in our existing sequence of poses for the subsequent animation phase. For a typical 3D motion sequence of about 300–500 animated poses, the optimization procedure takes less than one second to compute.

We have to stress that this algorithm does not solve the general camera calibration problem, but provides a specific solution aiming at reconstructing an approximate, but "plausible," camera model for a given image with user-selected 2D joints and a predefined 3D skeleton model, which is robust with respect to discrepancies between user-selected joints and the 3D data. Our algorithm might fail to converge to a proper solution in cases where the user-defined pose is too different from any pose within the selected 3D motion data sequence. In such cases, the user has the option to deactivate model joints, and to choose a specific, fixed field-of-view (which then influences the amount of foreshortening during animation). In some cases, such a user-controlled field-of-view can even be of advantage to emphasize perspective effects. On the other hand, our method proved to work quite robustly even on animal shapes that are quite different from a human shape (see Figure 7).

At the end of this step, we have computed the most likely camera projection matrix P and the best-matching 3D pose $\mathbb{X}_o$ which most closely approximates the user-selected joint positions. Based on this data, we initialize our algorithm for 2D shape animation.

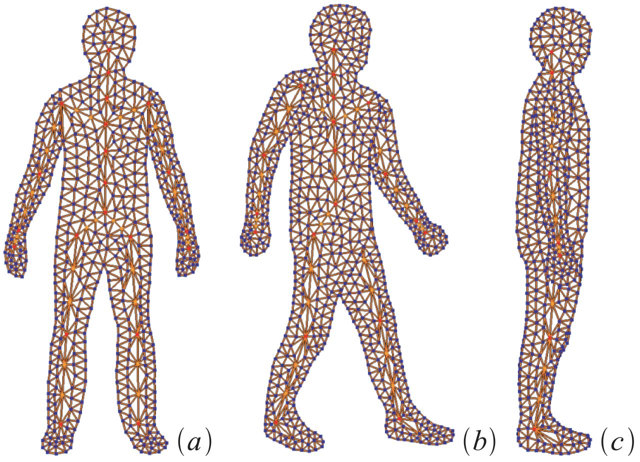

Fig. 3. Set of three shape templates generated by cycling in 45 degree steps around a person; (a) frontal view with one animation layer; (b) and (c) half-profile view and a side view with four layers each: foremost arm, body and front leg, and one for the leg and the arm at the back, respectively.

6. INITIAL MODEL-FITTING

To create convincing animations of a 2D character, we would like to deform its shape in a plausible way, while keeping the effort for generating animations on a minimal level. Our basic idea is to use a set of generic *shape templates* $\mathbb{T}$, which represents the topological changes of a generalized character model for different viewpoints of the camera (see Figure 3). These templates are fitted in a semiautomatic approach to the character. In our context, we define a shape template as a 2D nonmanifold triangle mesh which allows for the representation of different animation layers. For example, when animating characters from a side view, we have to consider different layers for separately moving parts of a body, for example, one layer for the foremost arm, one for the body and the foremost leg, and one for the remaining arm and leg, respectively (see Figure 3 (c)). Moreover, these layers cannot move independently, but have to be "stitched" together to convey the impression of a connected body when animated. Hence, each layer of a template $\mathbb{T}$ consists of a triangulated set of vertices representing the shape boundary and skeleton joints. The different layers of each template are connected by shared boundary vertices. Additional vertices are added inside each layer, which allows us to generate more realistic animations by "inflating" the shape templates prior to animation. This fact, however, is not important for the discussion of our initial model-fitting step in this section and will be explained in Section 7.

For a given image, a shape template can be automatically selected by exploiting the reconstructed extrinsic camera data R and $\mathbf{C}$ in relation to the best-matching model pose $\mathbb{X}_o$. In the following, we will explain how this generic shape can be fitted to the user-selected pose and employed to resolve occlusions for reconstructing a proper texture and alpha matte for each template layer.

For the deformation of $\mathbb{T}$, we use an as-rigid-as-possible (ARAP) shape manipulation technique, similar to Igarashi et al. [2005], which on the one hand allows for flexible deformations of the character's shape, and on the other, preserves the size and aspect ratios of the character's body to bound the distortion.

The ARAP algorithm transforms a shape template $\mathbb{T}$ (see Figure 4 (a)) into a deformed shape $\mathbb{D}$, while satisfying a set of vertex position constraints. To ensure an as-rigid-as-possible deformation, the algorithm consists of three steps which aim at preserving

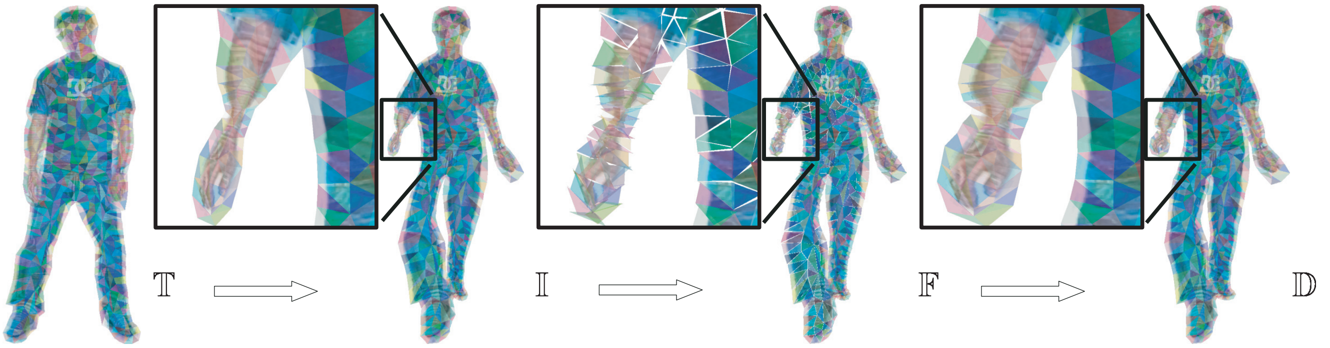

Fig. 4. The as-rigid-as-possible shape manipulation technique, used for computing deformed characters, consists of three steps: A triangulated shape $\mathbb{T}$ is converted into an intermediate mesh $\mathbb{I}$ which satisfies a set of vertex constraints. The original faces of $\mathbb{T}$ are rigidly fitted to the faces in $\mathbb{I}$, resulting in a disconnected mesh $\mathbb{F}$. A final averaging step then creates the desired deformed shape $\mathbb{D}$, which satisfies the joint constraints while preserving the rigidity of each triangle. Note the much more realistic shape deformation of $\mathbb{D}$ in comparison to $\mathbb{I}$, for example, around the legs and arms.

the shape of the original triangle faces $f \in \mathbb{T}$. In a first step, an intermediate shape $\mathbb{I}$ is computed for the given vertex constraints using, for example, a Laplace-based deformation [Sorkine et al. 2004] (see Figure 4 (b)). Since this step does not prevent an arbitrary scaling of the triangles $f \in \mathbb{T}$, a second scale-adjustment step rigidly fits each face f in a least-squares sense to the corresponding face in the intermediate shape, using translation and rotation only. This results in an unconnected set $\mathbb{F}$ of triangles (see Figure 4 (c)). This set is then converted into the final deformed shape $\mathbb{D}$ (see Figure 4 (d)) by computing the average of corresponding vertex positions in $\mathbb{F}$ (see, e.g., Igarashi et al. [2005] for details). Since all computations for the intermediate and fitted, as well as the final deformed shape, are either done on a per vertex or per triangle basis, this algorithm is fully compatible with our nonmanifold shape templates $\mathbb{T}$.

A deformed shape $\mathbb{D}$ can now easily be generated by specifying updated positions for inner-joint vertices of $\mathbb{T}$. To ensure that the limbs of a character are truly rigid during deformation, and that the shape deformation is mainly restricted to triangles located at joints, we place additional constraint vertices along each skeleton bone, and refine the template triangulation accordingly.

The remaining steps to generate a final shape used for animation then proceed as follows. The shape template for a given image is first fitted to the user-selected joint positions using a single ARAP deformation step $\mathbb{T} \rightarrow \mathbb{I} \rightarrow \mathbb{F} \rightarrow \mathbb{D}$. Depending on the mismatch between the contour of the template and the character in the input image, the boundaries of $\mathbb{D}$ are "snapped" to the character's silhouette using a 2D-snakes approach [Kass et al. 1987]. Those parts of the silhouette which do not contain enough information for a stable convergence, such as occluded regions of the body, can be manually improved by the user. This refined shape is defined as the final aligned shape $\mathbb{T}'$.

To generate textures for each animation layer, we decompose $\mathbb{T}'$ by a front-to-back depth-peeling algorithm. For each of these layers, including the background, we complete disoccluded texture regions using an interactive method for image completion [Pavić et al. 2006]. Poisson-matting [Sun et al. 2004] is used to extract an alpha matte for each layer.

The initial model-fitting is concluded by applying a final ARAP manipulation step $\mathbb{T}' \rightarrow \mathbb{I} \rightarrow \mathbb{F} \rightarrow \mathbb{T}_o$, transforming the character's image pose, as defined by the user-selected joints, into the best-matching 3D pose by using the projected joint positions $\mathsf{P}\mathbf{X}_i, \mathbf{X}_i \in \mathbb{X}_o$ as vertex constraints. The resulting shape yields the initial starting shape used for the subsequent animation.

7. ANIMATION

Using the ARAP technique of Igarashi et al. [2005], we could easily animate the textured shape template $\mathbb{T}_o$ by constraining the bone vertices to the 2D positions obtained by sequentially projecting the poses $\mathbb{X}_t$ from the motion dataset into the image plane. However, since ARAP is aiming at the rigid preservation of the original 2D triangle shapes, this will not lead to plausible deformations because perspective scaling and distortion effects are completely ignored. This leads to unrealistic distortions, such as thinning or thickening of the character's limbs when they are changing their orientation relative to the image plane (see Figure 5).

Our solution is to generalize the ARAP technique to an as-similar-as-possible (ASAP) technique. Here, we still perform a sequence of deformation steps $\mathbb{T}_o \rightarrow \mathbb{I} \rightarrow \mathbb{F} \rightarrow \mathbb{D}$ for every animation frame and pose $\mathbb{X}_t$, but instead of rigidly fitting the original triangles $f \in \mathbb{T}_o$ to the intermediate shape $\mathbb{I}$ to obtain $\mathbb{F}$, we estimate their perspective distortion, and fit the distorted triangles f' to $\mathbb{I}$. By this process, we eventually generate a deformed shape $\mathbb{D}$ whose faces are as *similar* as possible to the perspectively distorted triangles f'.

To estimate the perspective distortion of a triangle $f \in \mathbb{T}_o$, we exploit the 3D information given by the motion data. With each bone $\mathbf{b} = \mathbf{X}_i - \mathbf{X}_j$ defined by two neighboring joints in a pose $\mathbb{X}$, we associate a local coordinate frame L, which changes according to the bone's movement. This change of the bone's orientation provides the necessary information to compute the triangle's perspective distortion.

Let $\mathbf{b}_o$ be a bone's orientation in the initial pose $\mathbb{X}_o$. Its local frame L_o is defined by three vectors: $\mathbf{l}_x = (\mathbf{C} - \mathbf{X}_{i,o}) \times \mathbf{b}_o$, $\mathbf{l}_y = \mathbf{b}_o$, and $\mathbf{l}_z = \mathbf{l}_x \times \mathbf{l}_y$, with $\mathbf{C}$ being the camera center (see Section 5). Likewise, we define the local frame L_t for the same bone as it is oriented in the target pose $\mathbb{X}_t$.

We first consider the triangles in $\mathbb{T}_o$ which are uniquely associated with a single bone $\mathbf{b}_o$. The 2D vertices $\mathbf{v}$ of such a triangle are unprojected to 3D by mapping them to a point $\mathbf{V}_o$ on the plane spanned by the vectors, $\mathbf{l}_x$ and $\mathbf{l}_y$, of $\mathbf{b}_o$'s local frame L_o. In order to avoid a cardboard effect in the photo animation at grazing viewing angles (see Figure 5), we "inflate" the shape templates by adding

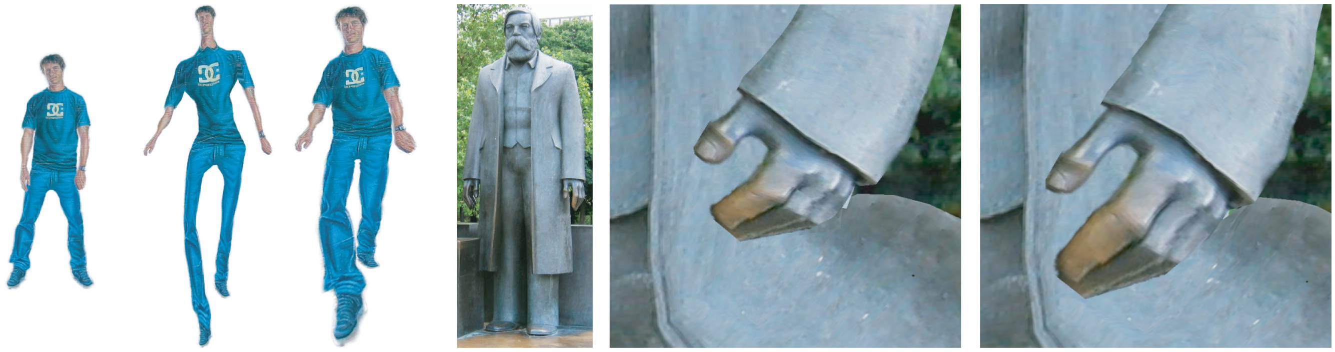

Fig. 5. When animating 2D shapes with 3D motion data, the original ARAP shape manipulation leads to unnatural results, since it cannot handle size changes and other perspective distortion effects. The left part of this figure shows the original Skater model, and an enlarged version after a few animation frames of forward motion without ARAP and with ASAP, our perspective triangle correction step. The ARAP approach tries to preserve the size of the original triangles, leading to significant thinning of the overall character. Our ASAP approach resolves this problem by recomputing corrected triangles. The righthand-side visualizes the difference of using simple planar shape templates versus inflated templates. When seating the originally standing Engels statue, the arms come to rest on the legs at a grazing viewing angle with respect to the viewer. With flat shape templates, this leads to a noticeable "cardboard" effect. In contrast, our inflated shape templates lead to a more natural deformation.

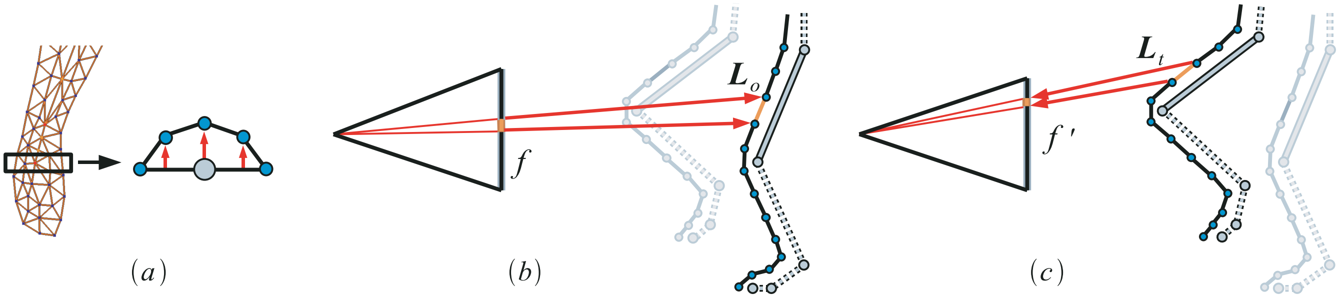

Fig. 6. Projective triangle correction step in 2D; (a) cross-section of the leftmost shape template in Figure 3. A curved surface is simulated by "inflating" the vertices of the shape templates along the normal direction of each bone's supporting plane. Each face $f \in \mathbb{T}_o$ is then projected from image space into the local coordinate system L_o of the corresponding bone in the initial pose $\mathbb{X}_o$; (b) then, a projectively deformed triangle f' is generated by updating $\mathsf{L}_o \rightarrow \mathsf{L}_t$, and reprojecting the corresponding triangle vertices back into the image (c).

an offset $\alpha \mathbf{l}_z$ to the 3D position $\mathbf{V}_o$, where the factor α is zero for boundary vertices of the template $\mathbb{T}_o$, and increases for vertices closer to the bone (see Figures 6 (a) and (b)). Then, $\mathbf{V}_o$ is expressed in local coordinates with respect to $\mathbf{b}_o$, that is, $\widetilde{\mathbf{V}}_o = \mathsf{L}_o^{-1}(\mathbf{V}_o - \mathbf{X}_{i,o})$.

For another frame $\mathbb{X}_t$ of the motion data, we obtain the 3D position of a particular vertex by transforming its local coordinates $\widetilde{\mathbf{V}}_o$ back to global coordinates $\mathbf{V}_t = \mathsf{L}_t \widetilde{\mathbf{V}}_o + \mathbf{X}_{i,t}$, and then projecting this point back to the image plane $\mathbf{v}' = \mathsf{P}\mathbf{V}_t$. Applying this procedure to all three vertices of a triangle f yields the correctly distorted triangle shape f', which we can use for the ASAP deformation.

For triangles near a skeleton joint which are not associated with a single bone, we apply the same procedure for each associated bone and then compute a weighted average position in 3D.

As an alternative to the original approach described in Igarashi et al. [2005], we simplify the rigid fitting step $\mathbb{I} \rightarrow \mathbb{F}$ by computing a closed form solution for the optimal rotation which minimizes the squared error $\sum_i \|\mathbf{p}_i - \mathbf{q}_i\|^2$ over the vertices $\mathbf{p}_i \in f'$ and the vertices $\mathbf{q}_i$ of the corresponding triangle in $\mathbb{I}$. First, we translate f' into the center of gravity of the vertices $\mathbf{q}_i$. Then we compute the optimal 2D rotation angle ψ for f' using Horn [1987]:

$$\widetilde{\mathsf{R}} = \sum_i (< p_i, q_i >, < p_i, q_i^{\perp} >), \quad (\cos\psi, \sin\psi) = \frac{\widetilde{\mathsf{R}}}{\|\widetilde{\mathsf{R}}\|} \ . \quad (4)$$

This fitting step is computed for each triangle, and the resulting deformed shape $\mathbb{F} \rightarrow \mathbb{D}$ is computed as in Igarashi et al. [2005] by averaging the unconnected triangle vertices.

To summarize, the final animation can now be generated by simply updating 2D joint positions $\mathbf{x}_i = \mathsf{P}\mathbf{X}_i$ for subsequent poses $\mathbb{X}_t$ from the user chosen 3D motion sequence. The corresponding skeleton vertices of the shape template are used as constraints to compute an intermediate shape $\mathbb{T}_o \rightarrow \mathbb{I}$. Projectively corrected triangles computed by the preceding algorithm are used to generate a final as-similar-as-possible deformed mesh $\mathbb{I} \rightarrow \mathbb{F} \rightarrow \mathbb{D}$. For rendering, we simply use textured OpenGL triangle meshes with enabled alpha blending. Triangle depths for proper occlusion handling can be derived directly from the corresponding 3D pose. This enables us to generate and view the resulting animation in real time.

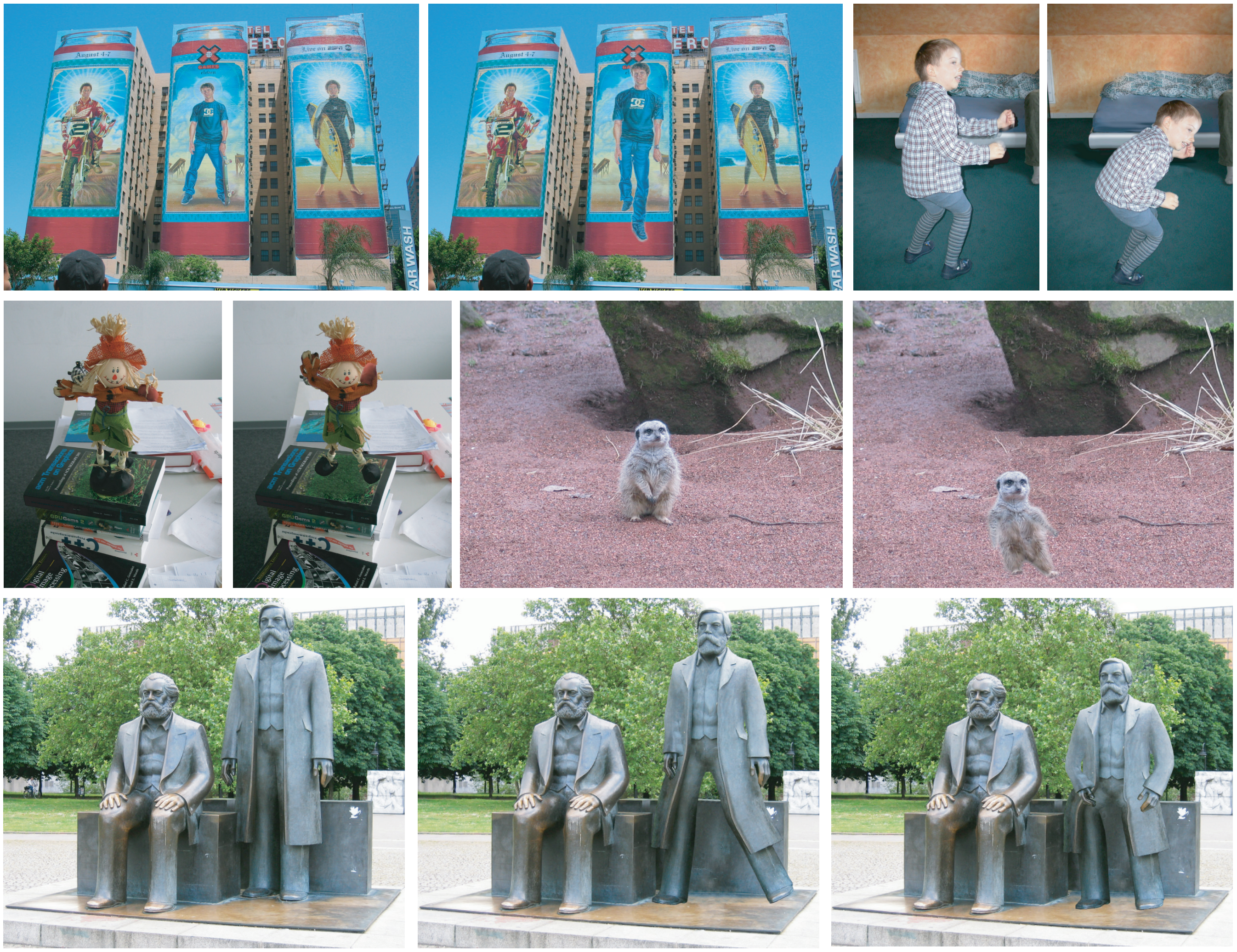

Fig. 7. Our method enables the animation of a variety of 2D characters with realistic 3D motion. In these examples, the respective lefthand-side of an image pair shows the original input picture, while the righthand-side is a frame from an animation generated with our system. In the first row, we animate the photograph of a Skateboarder painted on a wall with walking motion, and let Philipp perform the "Chicken-Dance." The middle row shows a Scarecrow model jumping and waving on an office desk, and a Meerkat dancing the Lambada. The last row shows a statue of Engels having a rest beside Marx. Please refer to the accompanying video available at http://www.rwth-graphics.de/downloads/CharAnim.avi for the full animations.

As a final note, we have to mention that new animations can be generated trivially by simply exchanging the 3D motion. All previously computed steps, such as the boundary snapping and texture generation, do not have to be recomputed. Since the camera and model pose can be computed on-the-fly, applying a different motion to a character is generally a matter of seconds.

8. RESULTS

Figures 1 and 7 show selected keyframes of image animations generated with our system. We animate a painting of Napoleon and the photograph of a painted Skater with walking motion, and let Philipp perform the "Chicken Dance." The image of a Scarecrow model is animated with jumping and waving motion, and a Meerkat dances the Lambada. Finally, we let a standing statue of Engels take a seat beside Marx. Please refer to the full animations shown in the accompanying video available at http://www.rwth-graphics.de/downloads/CharAnim.avi. Our results show that it is possible to animate a variety of images from photographs and paintings of people to animals with a wide range of 3D motions, although the actual image texture remains static and does not change with the motion. However, competing with "ground truth data", that is, a comparison to video sequences of moving persons, has explicitly not been the goal of this work. Rather, we see the strength of our method as the possibility of convincingly posing or animating *any* kind of character which has a structure roughly corresponding to the available 3D motion data.

Our set of predefined shape templates and the corresponding number of layers are shown in Figure 3. While we can generally use this set for animating frontal views of humans, we applied slight modifications to our standard shapes for animating the Meerkat, since it has significantly shorter legs in our example. For the Napoleon example, we simply removed the layer for the hindmost arm. However, creating a modified base shape is as easy as the joint selection and boundary snapping step during the initial model-fitting.

The user interaction times for all the examples shown in this article were between 15 to 30 minutes for the skeleton joint selection and the boundary snapping. As already mentioned in previous work on similar image processing tasks (e.g., Chuang et al. [2005]), the texture completion and alpha-matting turns out to be most time-consuming step. However, by using interactive methods, such as Pavić et al. [2006], results with good quality can be produced in about 10 to 20 minutes. Overall, even high-quality photo animations of complex images (including optional postprocessing of the image completion and matting results) generally do not take much longer than one hour to prepare.

9. CONCLUSION

In this article, we presented a complete, easy-to-use system for animating 2D images of arbitrary characters with 3D motion. We showed how simple user interaction, namely the selection of a few 2D joint positions, can be exploited to automatically reconstruct a geometrically plausible camera calibration and model pose from 3D motion data. We presented an initial model-fitting step using a generic set of shape templates to animate arbitrary human characters. Finally, we introduced an as-similar-as-possible shape deformation algorithm to deform these shapes in a projectively more correct manner, allowing us to generate still frames and animations of a large variety of characters from different types of 3D motion.

We believe (this is also mentioned in previous work in this area) that methods working solely on a single image have certain natural restrictions as to what kind of modifications, or even animations, can be achieved. One restriction of our technique is the fact that it obviously does not work for motions where the character changes its moving direction, or where it turns its head, for example. This would imply on-the-fly switching of the shape template, and even more importantly, resynthesizing occluded textures would be much more difficult. Nevertheless, in many application scenarios, there is just one single image available, hence, we have to do our best based on this restricted information. We think that our method provides a good basis for future work, since it identifies the involved problems and presents a first complete and flexible solution to this problem domain. In combination with other techniques mentioned in our related work section, these methods could eventually be integrated into a powerful toolkit for computer animation from single pictures.

Besides the aforementioned restrictions, there is quite a number of points which we would like to investigate in our future work. Methods for automatic pose estimation would reduce the amount of user interaction in cases where the 2D character is in a relatively common pose, such as standing or walking. Furthermore, we would like to generate smooth transitions from the user-selected pose to the best-matching pose of the 3D motion, or allow for transitions between different 3D motions. It would also be very interesting to integrate global effects such as shadows or reflections to improve the visual appearance of some scenes. Finally, a larger set of 3D motions would allow us to animate animals, or even plants. These issues provide an interesting basis for future research.

ACKNOWLEDGMENTS

This work would not have been possible without the help of Martin Habbecke, Jan Möbius, Darko Pavić, Arne Schmitz, and Volker Schönefeld. The painting of Napoleon (see Figure 1) is courtesy of WebMuseum at http://www.ibiblio.org/wm/. The motion data used in this project was obtained from http://mocap.cs.cmu.edu.

REFERENCES

AGARWAL, A. AND TRIGGS, B. 2006. Recovering 3d human pose from monocular images. *IEEE Trans. Pattern Anal. Mach. Intel. 28*, 1, 44–58.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-Rigid-As-Possible shape interpolation. In *Proceedings of the SIGGRAPH Conference*. 157–164.

BARRETT, W. A. AND CHENEY, A. S. 2002. Object-Based image editing. In *Proceedings of the SIGGRAPH Conference*. 777–784.

BREGLER, C., LOEB, L., CHUANG, E., AND DESHPANDE, H. 2002. Turning to the masters: Motion capturing cartoons. In *Proceedings of the SIGGRAPH Conference*. 399–407.

CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2005. Animating pictures with stochastic motion textures. *ACM Trans. Graph. 24*, 3, 853–860.

CMU GRAPHICS LAB MOTION CAPTURE DATABASE. 2007. http://mocap.cs.cmu.edu/.

CORRÊA, W. T., JENSEN, R. J., THAYER, C. E., AND FINKELSTEIN, A. 1998. Texture mapping for cel animation. In *Proceedings of the SIGGRAPH Conference*. 435–446.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Fragment-Based image completion. *ACM Trans. Graph. 22*, 3, 303–312.

EFROS, A. A. AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of the 7th International Conference on Computer Vision*. 1033–1038.

GAVRILA, D. M. 1999. The visual analysis of human movement: A survey. *Comput. Vision Image Understanding. 73*, 1, 82–98.

HARTLEY, R. AND ZISSERMAN, A. 2003. *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, New York.

HORN, B. 1987. Closed form solutions of absolute orientation using unit quaternions. *J. Optical Soci. America 4*, 4 (Apr.), 629–642.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-Rigid-As-Possible shape manipulation. *ACM Trans. Graph. 24*, 3, 1134–1141.

JOJIC, N. AND FREY, B. J. 2001. Learning flexible sprites in video layers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 199–206.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1987. Snakes: Active contour models. *Int. J. Comput. Vis.* 321–331.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the SIGGRAPH Conference*. 491–500.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the SIGGRAPH Conference*. 165–172.

LIU, C., TORRALBA, A. B., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. 2005. Motion magnification. *ACM Trans. Graph. 24*, 3, 519–526.

MOESLUND, T. B. AND GRANUM, E. 2001. A survey of computer vision-based human motion capture. *Comput. Vision Image Understanding. 81*, 3, 231–268.

MORI, G., BERG, A., EFROS, A., EDEN, A., AND MALIK, J. 2004. Video based motion synthesis by splicing and morphing. Tech. Rep. UCB//CSD-04-1337, University of California, Berkeley. June.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-Based modeling and photo editing. In *Proceedings of the SIGGRAPH Conference*. 433–442.

ONO, Y., CHEN, B.-Y., AND NISHITA, T. 2004. 3D character model creation from cel animation. In *IEEE Cyberworlds*. 210–215.

PARAMESWARAN, V. AND CHELLAPPA, R. 2004. View independent human body pose estimation from a single perspective image. In

Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 16–22.

PAVIĆ, D., SCHÖNEFELD, V., AND KOBBELT, L. 2006. Interactive image completion with perspective correction. *Vis. Comput. 22*, 9, 671–681.

RADEMACHER, P. 1999. View-Dependent geometry. In *Proceedings of the SIGGRAPH Conference*. 439–446.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the Symposium on Geometry Processing*. 179–188.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. *ACM Trans. Graph. 23*, 3, 315–321.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Trans. Graph. 24*, 3, 861–868.

TAYLOR, C. J. 2000. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1677–1684.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Trans. Graph. 23*, 3, 424–431.

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R., AND FITZGIBBON, A. 2000. Bundle adjustment—A modern synthesis. In *Vision Algorithms: Theory and Practice*, B. Triggs et al., Eds. Lecture Notes in Computer Science. Springer Verlag, 298–375.

Received June 2006; accepted October 2006