

# Three-Dimensional Proxies for Hand-Drawn Characters

EAKTA JAIN and YASER SHEIKH

Carnegie Mellon University

and

MOSHE MAHLER and JESSICA HODGINS

Carnegie Mellon University and Disney Research Pittsburgh

Drawing shapes by hand and manipulating computer-generated objects are the two dominant forms of animation. Though each medium has its own advantages, the techniques developed for one medium are not easily leveraged in the other medium because hand animation is two-dimensional, and inferring the third dimension is mathematically ambiguous. A second challenge is that the character is a consistent three-dimensional (3D) object in computer animation while hand animators introduce geometric inconsistencies in the two-dimensional (2D) shapes to better convey a character's emotional state and personality. In this work, we identify 3D proxies to connect hand-drawn animation and 3D computer animation. We present an integrated approach to generate three levels of 3D proxies: single-points, polygonal shapes, and a full joint hierarchy. We demonstrate how this approach enables one medium to take advantage of techniques developed for the other; for example, 3D physical simulation is used to create clothes for a hand-animated character, and a traditionally trained animator is able to influence the performance of a 3D character while drawing with paper and pencil.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—*Animation*; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms: Algorithms

Additional Key Words and Phrases: Hand animation, 3D simulation

## ACM Reference Format:

Jain, E., Sheikh, Y., Mahler, M., and Hodgins, J. 2012. Three-dimensional proxies for hand-drawn characters. *ACM Trans. Graph.* 31, 1, Article 8 (January 2012), 16 pages.

DOI = 10.1145/2077341.2077349

<http://doi.acm.org/10.1145/2077341.2077349>

Disney Research for summer funding. Partial funding was also provided by NSF CCF-0811450 and NSF IIS-0916272.

Authors' addresses: E. Jain (corresponding author), Y. Sheikh, Computer Science Department, Carnegie Mellon University; email: [ejain@cs.cmu.edu](mailto:ejain@cs.cmu.edu); M. Mahler, and J. Hodgins, Computer Science Department, Carnegie Mellon University and Disney Research Pittsburgh, PA.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2012 ACM 0730-0301/2012/01-ART8 \$10.00

DOI 10.1145/2077341.2077349

<http://doi.acm.org/10.1145/2077341.2077349>

## 1. INTRODUCTION

Animators adopt many different media to make their characters move and emote; some favor sketching with a pencil, others like to manipulate armatures, sculpt clay, move paper cutouts, or pose 3D computer-generated skeletal rigs. Each medium has advantages and disadvantages, and requires that the artist develop a medium-specific set of skills to create a visually compelling end-product [Lasseter 1994]. In this work, we consider the differences between 2D hand-drawn animation and 3D computer-generated animation.

Sketching the basic idea with a pencil is quick, but art directing hand-drawn animations is cumbersome. For example, if the director wants a change in how the lead character's skirt swishes, the animator has to redraw the skirt for every frame. In contrast, the medium of 3D computer animation makes many components easily modifiable; the clothes can be resimulated, the viewing camera can be moved, and the lighting can be changed with little work on the part of the artist. This modifiability comes at the cost of increased initial effort to set up the scene and dynamic simulations, yet it enables 3D computer animation to be more easily art-directed than traditional hand-drawn animation.

The skills involved in hand-drawn animation and 3D computer animation are different too. Traditional animators are trained to communicate the movement of the character through pencil strokes [Johnston and Thomas 1995]. In contrast, 3D artists are skilled at using computer software and at manipulating curves, control points, and inverse kinematics handles to construct a performance for their character.

These observations motivate the question addressed in this article: how can we connect these two different media so as to leverage the programmability of 3D computer animation in the hand-drawn medium, and the unique talent pool of traditional animators in the 3D computer-generated medium? The challenge here is that the elements of 3D computer animation and 2D hand-drawn animation are fundamentally incompatible; the character exists as a consistent 3D object in the former, while 2D artists routinely introduce inconsistencies in their drawings to add expressiveness to the character [Rademacher 1999]. There may, in fact, be no consistent geometry that can be used to accurately describe a hand-drawn character from all points of view, or to define the imaginary camera used by the artist to create her drawing.

We propose that hand-drawn characters be represented by 3D proxies with different levels of detail. In Figure 1, we arrange the 3D proxies that can be created for a given hand-drawn animation according to increasing level of detail. For example, if we wanted to attach a simulated balloon onto the wrist of a hand-drawn character (Figure 1, top row), the level of detail for the 3D proxy would be a plausible 3D trajectory for a chosen point (wrist marker) on the 2D hand-drawn character: the single-point 3D proxy. Attempting to reconstruct the full 3D pose, minimizing an objective function with rigidity constraints, or enforcing good ground contact models







3D proxy	Degrees of freedom	Function
 <p>Single-point</p>	$3N$ $N = \text{number of attachment points}$ (typically, $N = 1, 2, 3$ )	 <p>Attachment point</p>
 <p>3D Polygonal shapes</p>	$5V$ $V = \text{number of tapered cylinders}$ (typically, $V = 1 \text{ to } 25$ )	 <p>Collision volume</p>
 <p>Joint hierarchy</p>	$3J$ $J = \text{number of joints}$ (typically, $J = 20$ )	 <p>Skinned, lit, rendered from any viewpoint</p>

Fig. 1. Three-dimensional proxies for hand-drawn characters are arranged in increasing order of number of degrees of freedom. The first level is the single-point proxy. The next level is approximate cylindrical models for those body parts which interact with 3D scene elements. The third level is a hierarchical joint model, which can be rigged, skinned, lit, and placed in a 3D scene.

would not only be excessive, but might minimize extraneous error terms at the expense of introducing error in the trajectory that will be visible in the final animation.

The next higher level of detail would be an approximate 3D model composed of tapered cylinders and spheres. These 3D proxies are well-suited to drive simulated effects like cloth and fluids, without having to fit exactly to the hand-drawn pencil strokes, because they will only provide the driving signal and will never be directly rendered. Here, the objective is to track points of contact precisely, and approximate the rest of the character with polygonal shapes, for example, precisely tracking points on the wrist so that the scarves appear to be attached to the ballerina, but approximating the arms as tapered cylinders (Figure 1, middle row).

Finally, transferring the style of the hand-drawn animation onto a hierarchical 3D joint model would allow traditionally trained animators to control a classic skeleton-based 3D character; animators who think in terms of pencil lines would be able to puppeteer 3D characters that can also be rigged, skinned, lit, and accessorized with simulated clothes (Figure 1, bottom row). Here, smoothness and naturalness of motion take on more importance than accurately tracking image pixels.

In this article, we present an integrated approach to generate three levels of 3D proxies for a hand-animated character: the single-point proxy, the tapered cylinder model, and the hierarchical joint model. We frame the solution as a least squares minimization and show that by reformulating the terms of the minimization according to the level of detail desired in the 3D proxy, we can generate these proxies for a hand-drawn character. Earlier versions of this work were presented by Jain and colleagues [2009, 2010]. Here, we generalize the technique and more extensively evaluate the various underlying assumptions.

We have made certain design choices with the standard animation pipeline in mind; when choosing inputs for the system, we employ user interaction that can be reasonably integrated into the traditional animation workflow. We present results on a variety of hand-drawn animated characters, ranging from stick figures to a ballerina with human shaped limbs, and evaluate the system through synthetic tests and hand-animations that stretch the underlying assumptions.

## 2. RELATED WORK

We discuss past work on the use of 3D representations for 2D objects in the Computer Graphics (CG) context and the techniques used to generate them. There has also been a great deal of research in reconstructing 3D human pose from images and we briefly survey that body of work in the computer vision community.

### 2.1 Computer Graphics

One of the earliest forms of 3D proxy for a hand-drawn scene element was used in the movie *Spirit*: 3D models of hand-animated horses were carefully made to match the artist's lines, and shots switched seamlessly between the hand-drawn elements and the 3D elements depending on whether the characters were seen in close-up or long shots [Cooper 2002].

Petrovic and colleagues [2000] create a blobby 3D proxy to generate ray-traced shadows for a 2D character by inflating the 2D shape along the third dimension. Their main contribution is a graphical interface that allows the user to specify the relative depths of every scene element that needs to be inflated. Correa and colleagues [1998] warp a 3D mesh model to match the artist-drawn shape; the mesh model can then be relit and textured and rendered to fill in the insides of the artist's hand-drawn shape with an intricate pattern. Johnston [2002] bypasses a full 3D proxy in favor of only interpolating normals, a 2.5D proxy that is sufficient to integrate 3D lighting with a hand drawing. All these methods are designed to work with blobby shapes and do not maintain articulation constraints.

There are also several previous techniques for lifting hand-drawn articulated characters from 2D to 3D. All of these techniques deal with various ways to resolve the forward-backward depth ambiguity. For example, Davis and colleagues [2003] sort the multiple 3D interpretations of a 2D pose according to joint angle constraints and other heuristics, but leave the final selection to the user. Their goal is to simplify the process of creating a 3D pose; the artist needs to know how to draw and mouse-click, but need not be trained in rigging and manipulating inverse kinematics handles. The generated 3D pose can be thought of as a single-point 3D proxy because it consists of 3D markers at joint locations. It would be time-consuming to generate a 3D proxy by their method for a full animation because of the nature of the user input involved.

Wei and colleagues [2010] generate physically realistic human motion from uncalibrated monocular data. Their method relies heavily on the assumption that the human skeleton is comprised of rigid

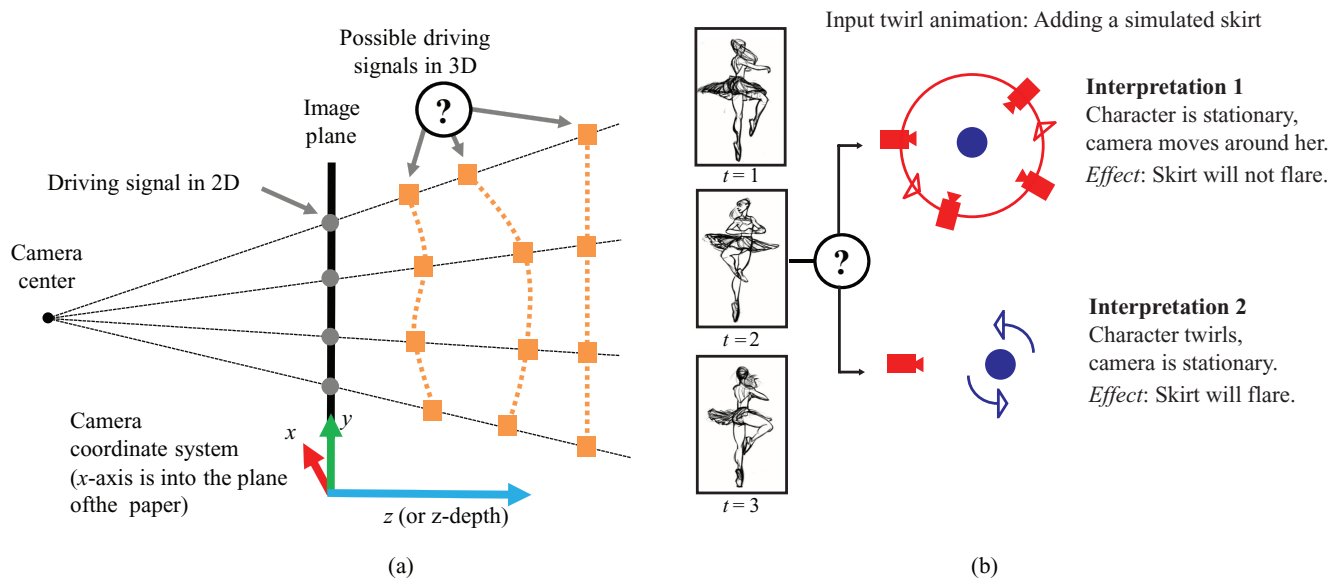


Fig. 2. Ambiguities in creating a 3D proxy. (a) Depth ambiguity: Multiple 3D trajectories can yield the same 2D projected path. (b) Composite motion ambiguity: The motion of the camera can not be disambiguated from the motion of the character if we are only given the image plane information.

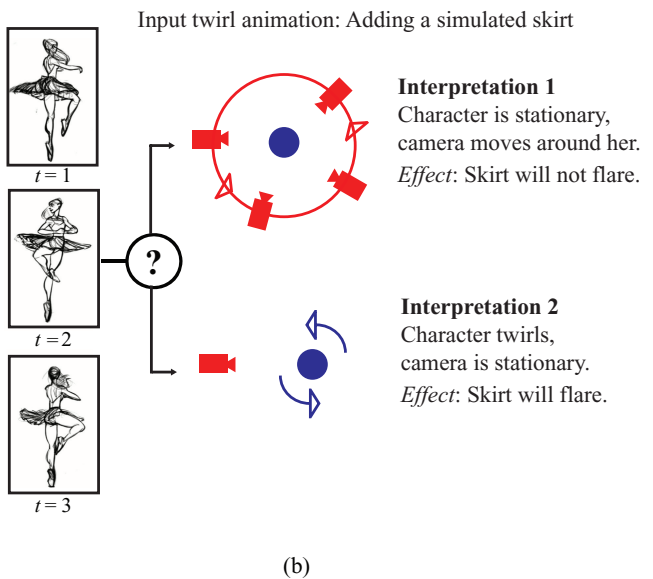
bones and that there are various symmetries in the human body. Newtonian physics, friction, and contact forces inform other terms in their optimization, but need not be valid priors for traditional hand-drawn animation.

In addition to the work on 3D interpretations of a given hand-drawing, there has also been research on how a 2.5D popup may be used. Sykora and colleagues [2010] employ user-specified depth inequalities to generate a 2.5D popup that may be used for layering, shading, and generating stereoscopic views. Rivers and colleagues [2010] interpolate different views based on an underlying 2.5D model, so that a hand-drawn shape can be seen from a novel viewpoint.

Computer graphics techniques have also been used to create background scenery, either in the form of 2D paintings manipulated to look three dimensional [Wood et al. 1997; Robertson 1998], or as a 3D scene, as in *Tarzan's* Deep Canvas [Daniels 1999]. These approaches do not allow for physical interaction between the hand-drawn elements and the 3D elements; the CG background can be composited with the hand-drawn foreground, but does not interact with it, for example, there are no dynamically simulated ripples when *Tarzan* steps into a pool of water. Our work addresses the challenge of connecting a traditionally animated character with 3D CG elements by enabling the character to drive the motion of the 3D scene elements via its 3D proxy.

## 2.2 Computer Vision

The recovery of 3D human pose from images has been studied in the computer vision community for over three decades (see, for example, Moeslund and Granum [2006] and Forsyth et al. [2005]). At a high level, these methods look for a 3D pose which minimizes the geometric projection error, and is constrained by priors about the way humans are proportioned and how humans move; these priors include limits on joint angles [Sminchisescu and Triggs 2003; Herda et al. 2004], physical models of the body [Rosenhahn et al. 2007b], foot plants as a constraint [Rosenhahn et al. 2008], and known limb lengths [Lee and Chen 1985; Taylor 2000]. Sidenbladh and colleagues [2000] and Rosenhahn et al. [2007a] further



applied smoothness constraints across a video sequence. Articulation constraints, to ensure that limbs must remain connected at joints, have also been used in a number of approaches [Bregler and Malik 1998; Wu et al. 2003; Demirdjian et al. 2003]. Recently, dimensionality reduction methods, which rely on motion capture data to learn mappings, have become popular [Sidenbladh et al. 2002; Grochow et al. 2004; Urtasun et al. 2006]. Along with these generative approaches, a number of discriminative approaches have also been proposed. These methods learn regression functions to link appearance features to 3D structure [Elgammal and Lee 2004; Sminchisescu et al. 2005; Agarwal and Triggs 2006; Ramanan et al. 2004; Bourdev and Malik 2009].

While researchers have demonstrated that a variety of priors are useful to resolve the forward-backward depth ambiguity, the most important assumption made in all these approaches is that the input data has a true and consistent 3D interpretation. In our domain, characters are hand-drawn, rather than being recorded via physical cameras, and talented animators often purposely violate the constraints of a skeleton-based model.

## 3. APPROACH

The frames that are drawn by the artist contain a projected view of the animated character as seen from a stationary or moving camera. As a result, there are two types of ambiguity in creating a 3D proxy: the depth ambiguity and the ambiguity created by the composition of the character motion and the camera motion (illustrated in Figure 2). The depth ambiguity occurs because multiple 3D points can project to the same 2D point (Figure 2(a)). The composite motion ambiguity occurs because the hand-drawn frames do not contain sufficient information to disambiguate the motion of the camera from the motion of the character. Figure 2(b) illustrates the camera-character motion ambiguity. For the purpose of articulated pose reconstruction, Interpretation 1 and Interpretation 2 are equivalent. However, when placing the character in a 3D virtual world, choosing the correct interpretation is essential or the skirt will not have the correct dynamic motion.

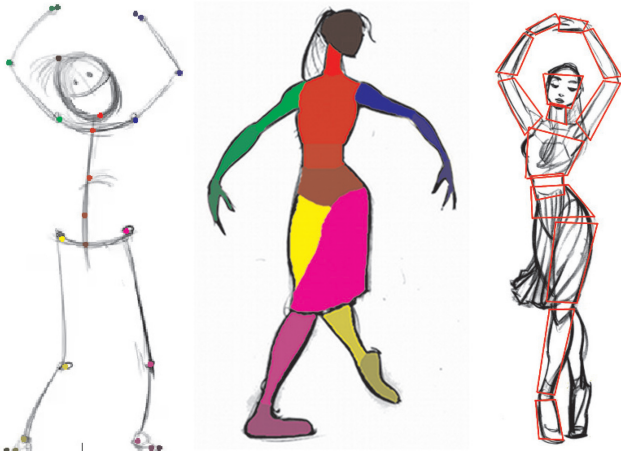


Fig. 3. Annotation: The user annotates the drawing with markers for joint locations (left). Color segmentation of the different body parts for the fully-fleshed characters (middle). Approximate bounding boxes, shown in red, for the limbs (right). The bounding boxes for the right leg have been omitted for clarity.

In this section, we will discuss how we preprocess raw data, estimate the camera that the artist might have imagined, and resolve the depth ambiguity for each of the three levels of 3D proxies: the single-point proxy, the tapered cylinder model, and the hierarchical joint model.

### 3.1 User Input

Because automatically tracking body parts in hand drawings is noisy, we ask a user to provide annotation of the hand-drawn animation. We also ask the user to select a motion capture segment to aid in resolving the depth and camera-character motion ambiguity. The user also provides an initialization that is used to estimate the artist's camera.

**3.1.1 Annotating the Hand-Drawings.** We ask a user (who can be a lay person) to specify the skeleton of the hand-drawn character with  $N$  virtual markers and the approximate bounding box for every limb. This annotation is done for each frame of the input animation. The user also provides a segmentation of the different body parts by color coding the interior of the hand-drawn figure (Figure 3). These user inputs, shown in Figure 3, are designed to fit into the traditional 2D animation workflow [Culhane 1990; Johnston and Thomas 1995]. The workflow begins with the keyframe animator, who roughs out six to eight frames for each second of animation. The cleanup and inbetweening artist cleans up the rough lines and draws the intermediate frames. At this stage, the virtual markers and bounding boxes can be marked easily without requiring significant additional effort. Then, the drawings are transferred to the ink and paint department where the colors are painted in. The color segmentation user input can be obtained as part of the painting process.

For each frame  $i$ , the user-specified virtual markers on the hand-drawn frames are denoted  $\tilde{\mathbf{x}}_i = [\tilde{x}_1, \tilde{y}_1, \tilde{x}_2, \tilde{y}_2, \dots, \tilde{x}_N, \tilde{y}_N]^T$ .

**3.1.2 Selecting a Motion Capture Segment.** The user selects a motion capture segment that has a similar sequence of actions as the hand-drawn sequence. The 3D poses in this motion capture segment will be used to resolve the depth ambiguity. The selection of the motion capture segment also helps resolve the composite camera-

character motion ambiguity (Figure 2(b)); the system assumes that the root of the character moves according to the motion capture segment, and the remaining motion is camera motion. We therefore refer to this segment as a motion prior. This user input may be obtained from either the keyframe or cleanup animator through an interface that allows her to browse a motion capture database.

The motion prior can differ from the hand-animation in timing because we preprocess the segment via the Dynamic Time Warp algorithm [Sakoe and Chiba 1990; Ellis 2003]. We denote the time-warped segment  $\tilde{\mathbf{X}}$ . For a frame  $i$ , the 3D marker positions for the motion prior poses are  $\tilde{\mathbf{X}}_i = [\tilde{X}_1, \tilde{Y}_1, \tilde{Z}_1, 1, \tilde{X}_2, \tilde{Y}_2, \tilde{Z}_2, 1, \dots, \tilde{X}_N, \tilde{Y}_N, \tilde{Z}_N, 1]^T$ , expressed in homogeneous world coordinates. This sequence of poses could be a sequence produced by blending, interpolating, or editing poses from a database of motion capture or key frame motion.

**3.1.3 Rotating a Motion Prior Pose.** As a first approximation to the camera imagined by the artist, we ask the user to specify an orthographic camera  $\mathbf{R}_{2 \times 3}$ . Because roll can be reasonably assumed to be zero, this camera is characterized by two degrees of freedom: azimuth angle and elevation. The user specifies these angles through a graphical interface that allows the motion prior pose to be rotated until its orthographic projection visually matches the hand-drawing. The rotation angles are used to compute  $\mathbf{R}_{2 \times 3}$ , an orthographic initialization for the artist's camera in the first frame of the animation. The estimation of camera motion is described in Section 3.3.

## 3.2 Preprocessing

When transferring the style of the hand-drawn animation onto a hierarchical 3D skeleton, there is a trade-off between tracking the artist's lines precisely and generating smooth, natural 3D motion. This trade-off exists because talented hand-animators purposefully violate the rigidity of the human skeleton to convey emotion via squash and stretch. A hierarchical joint skeleton does not provide the same affordances; thus, tracking a squashed arm accurately might result in unnatural movement for the elbow joint in the 3D proxy.

We introduce a pose descriptor to extract the pose of the hand-drawn character, while filtering out changes in limb length. This pose descriptor quantitatively describes a 2D hand-drawn pose, and is translation and scale invariant. The intuition here is that a character can be in the same pose at different locations, and two characters can have the same pose even if their relative limb lengths are different. In this preprocessing step, we modify the motion prior poses to match the pose descriptors for the hand-drawn poses.

For a given 2D pose, the descriptor starts at the root (which is the pelvis) and travels every hierarchical limb chain. For every link in the chain, we determine the position vector of the child marker in a coordinate frame fixed to its parent. As illustrated in Figure 4, the position vector for the wrist would be calculated with respect to a coordinate frame fixed to the elbow. The reference orientation for this coordinate frame can be absolute (i.e., oriented along the  $x$ -axis of the world coordinate frame), or relative (i.e., oriented along the corresponding limb, in this case, *right radius*). The pose descriptor  $\mathbf{P}$  for a given pose would be the vector of polar angles for the position vectors of  $K$  virtual markers of the skeletal model

$$\mathbf{P} = [\theta_1, \theta_2, \dots, \theta_K]^T, \quad (1)$$

where  $K$  is the number of limbs that are needed to characterize the pose.

We first compute the pose descriptors for the hand drawings. Then the 3D motion prior poses  $\tilde{\mathbf{X}}_i$  are projected to two dimensions with

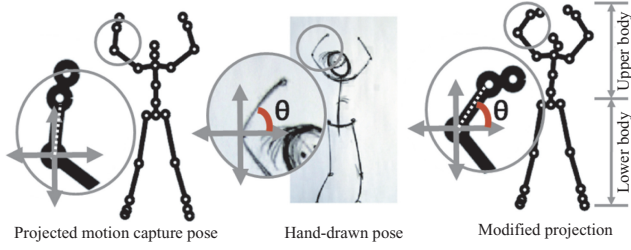


Fig. 4. Preprocessing: The pose descriptor consists of in-the-image-plane angles for every limb segment. The limb segments of the projected motion prior pose are modified to match the pose descriptor for the hand-drawn pose via planar rotation.

the camera approximation  $\mathbf{R}$ . The matrix  $\mathbf{R}$  is computed by taking the Kronecker product,  $\mathbf{R} = \mathbf{I}_N \otimes \mathbf{R}_{2 \times 3}$ . The projected motion prior poses are shown in Figure 4 (far left), where  $\tilde{\mathbf{x}}_{2D_i} = \mathbf{R}\tilde{\mathbf{X}}_i$ .

The character model is divided into “upper body”, which consists of the hierarchical chains containing the two arms and the head, and “lower body”, which consists of the limb chains involving the legs (Figure 4). We start modifying the projected motion prior pose at the pelvis and work out along each hierarchical chain of the upper body. Each limb segment of the projected motion prior pose is rotated in the image plane so that the in-plane polar angle is the same as the desired pose descriptor, that is, the corresponding polar angle in the hand-drawn pose (Figure 4, insets).

The modified 2D motion prior pose,  $\tilde{\mathbf{x}}_i^m$ , is illustrated in Figure 4 (far right). The lower body is not modified in order to transfer the ground contacts of the motion capture data onto the 3D joint hierarchy.

### 3.3 Camera Estimation

As illustrated in Figure 2(b), it is essential to resolve the camera-character motion ambiguity to place a 3D proxy in a virtual 3D world. However, the hand-drawings alone do not contain sufficient information to disambiguate the motion of the camera from the motion of the character. We resolve this ambiguity by registering the poses of the time-warped motion prior  $\tilde{\mathbf{X}}_i$  with the hand-drawn poses  $\tilde{\mathbf{x}}_i$ . The underlying assumption is that the root of the character moves according to the motion prior, any movement in the hand-drawn markers over and above the movement of the motion prior poses is attributed to camera movement.

For each frame  $i$ , we estimate a projection matrix  $\mathbf{M}_i$  that minimizes the geometric projection error,  $e_g$ ,

$$e_g = \sum_{t=-K/2}^{K/2} \|\tilde{\mathbf{x}}_{i+t} - \mathbf{x}_{i+t}^{proj}\|,$$

where  $\mathbf{x}_{i+t}^{proj} \cong \mathbf{M}_i \tilde{\mathbf{X}}_{i+t}$ . We compute across a moving window around the frame  $i$  to increase robustness to noise.

In addition to minimizing projection error, we also want  $\mathbf{M}_i$  to characterize a physically realizable camera that can render 3D elements: skew and tilt are set to zero, the scale factors are computed from the image resolution, and the focal length is prespecified. This formulation is similar to Hornung and colleagues [2007] and Petrovic and colleagues [2000]. As a result, only the external parameters for  $\mathbf{M}_i$  remain to be estimated: roll, pitch, yaw, and the location of the center. We denote the external parameters  $\rho(i) = (\theta_x(i), \theta_y(i), \theta_z(i), t_x(i), t_y(i), t_z(i))^T$ .

Constraints on the external parameters are that the renderable camera should be above ground level,  $e_l = (t_z - \mu)$ , roll orientation should be minimum,  $e_o = |\theta_y|$ , and the camera should move

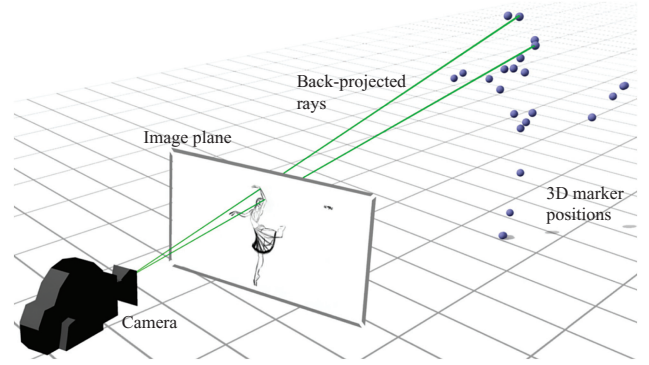


Fig. 5. Attachment points: Markers on the hand drawing are back-projected. Depth is obtained from the motion prior. The blue 3D markers illustrate what the solution looks like for a chosen frame.

smoothly,  $e_s = \|\rho(i) - \rho(i-1)\|$ . Therefore, we estimate  $\rho^*(i)$  such that

$$\rho^*(i) = \underset{\rho}{\operatorname{argmin}} (\omega_1 e_g + \omega_2 e_l + \omega_3 e_o + \omega_4 e_s), \quad (2)$$

where  $\omega_1, \omega_2, \omega_3$  and  $\omega_4$  are the associated weights. In practice,  $\omega_1 = 1, \omega_2 = \omega_3 = \omega_4 = 0.0005$ , and  $\mu = 1$ . We use a nonlinear optimizer, with the roll initialized to zero, the yaw and pitch set to the user-provided orthographic initialization, and the location of the center initialized approximately so that the first motion capture pose fits inside the image plane.

### 3.4 Detail-Dependent Minimization for 3D Proxy

Figure 1 illustrates the levels of 3D proxies for a hand-drawn character. We generate every 3D proxy by minimizing a weighted sum of three terms. Depending on the function of the 3D proxy, the individual terms are reformulated to best reflect the properties desired in the solution.

The first term is called the *input-match* term  $e_a$  and causes the 3D proxy to follow the hand-animation. Because the artist provides only the perspective drawing of the character, we need to infer the missing depth information. The second error term is the *motion prior*,  $e_m$ , which provides a data-driven prior for depth from motion capture data. The third term is the *regularization term*,  $e_r$ . These terms can be related to the formulation of 3D pose reconstruction in the computer vision literature: the input-match term,  $e_a$ , is analogous to the geometric projection error, the motion prior,  $e_m$ , is analogous to the data-driven or physics priors, and the regularization term,  $e_r$ , is analogous to temporal smoothing.

**3.4.1 Attachment Points.** To attach a 3D element to a hand-drawn character (for example, a simulated balloon onto the wrist of a hand-drawn figure), the level of detail for the proxy is plausible 3D trajectories for the attachment points. These 3D trajectories must achieve perfect image plane alignment with the hand-drawn character; only then will the 3D element appear to be convincingly attached to the artist-drawn character. Figure 5 illustrates the solution for one frame geometrically.

For each frame  $i$ ,  $\mathbf{M}_i$  is the projection operator and the 3D position of each marker  $j$  is denoted  $\mathbf{X}_{ij}^w = [X_{ij}^w, Y_{ij}^w, Z_{ij}^w, 1]^T$  in homogeneous world coordinates. Then, perfect image plane alignment is achieved by minimizing the input-match error  $e_a^{ij}$ ,

$$e_a^{ij} = \|\tilde{\mathbf{x}}_{ij} - \mathbf{x}_{ij}^{proj}\|, \quad (3)$$

where  $\tilde{\mathbf{x}}_{ij}$  are the virtual markers on the hand-drawing, and  $\mathbf{x}_{ij}^{proj}$  are the projections of the corresponding markers of the 3D proxy,

$$\mathbf{x}_{ij}^{proj} \cong \mathbf{M}_i \tilde{\mathbf{X}}_{ij}^w.$$

There can be infinitely many solutions which minimize the input-match error in Eq. (3). The motion prior term  $e_m^{ij}$  resolves this ambiguity by pulling the z-depth for each marker as close as possible to the corresponding value for the motion prior poses,  $\tilde{\mathbf{X}}$ . For the  $i^{th}$  frame,

$$e_m^{ij} = \|\mathbf{m}_3^T \tilde{\mathbf{X}}_{ij} - \mathbf{m}_3^T \mathbf{X}_{ij}^w\| \quad \forall j = 1, \dots, N. \quad (4)$$

The regularization term  $e_r^{ij}$  keeps the final trajectory temporally coherent. For each marker  $j$ ,

$$e_r^{ij} = \|\mathbf{X}_{ij}^w - \mathbf{X}_{(i+1)j}^w\|. \quad (5)$$

Finally, normalization constraints fix the scale factor in homogeneous coordinates to unity.

$$[0, 0, 0, 1] \mathbf{X}_{ij}^w = 1 \quad \forall j = 1, \dots, N \quad (6)$$

The Appendix contains details for how these terms are composed into a linear system, which can be solved in closed form.

**3.4.2 Collision Volumes.** To create believable interaction between the hand-drawn character and 3D objects, the proxy consists of 3D polygonal shapes that track the hand-drawn lines. Example interactions include splashes when the hand-drawn figure collides with simulated water, or a simulated scarf wrapping around the waist of the hand-drawn character. Because the 3D proxy only provides collision volumes and is never directly rendered, the polygonal shapes need not conform exactly to the hand-drawn lines. The three error terms,  $e_a$ ,  $e_m$ , and  $e_r$ , are reformulated to reflect these new constraints. The polygonal shapes used here are tapered cylinders for the arms, legs, and torso, and spheres for the joints, but other shapes would be feasible too.

The 3D polygonal shapes are generated by first computing the 3D positions of the end points of the cylinders. Here, as we explain the details for one limb, we will drop indices for clarity. Figure 6 shows the 3D positions,  $\mathbf{X}_A$  and  $\mathbf{X}_B$ , of the upper leg markers. This computation is identical to Section 3.4.1.

A tapered cylinder collision volume is completely characterized by the direction of its cylindrical axis, its height, and the radii of each face. The direction of the cylindrical axis  $\vec{a}\vec{x}$  and the height of the cylinder  $h$  are determined from the end point positions,  $\mathbf{X}_A$  and  $\mathbf{X}_B$  (Figure 6, Inset A). The radii of the two faces,  $r_1$  and  $r_2$ , depend on the thickness of the limb, as drawn by the artist. A simple algorithm approximates the artist-drawn limb with a bounding box, shown in Figure 6 as the quadrilateral  $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ . This algorithm identifies the artist-sketched limb outline by locating the first black pixel in the direction perpendicular to the line joining the virtual markers for that limb. Overlap between body parts will confuse this algorithm. When the arm crosses the torso, for example, the algorithm incorrectly marks the boundary of the arm as the torso boundary. This overlap may never occur for some actions (jumping jacks, for example), and may occur for other actions such as a dance. In the ballet example, an arm overlaps with the torso for 43 of the 200 frames. Such overlap cases are corrected by the user through an annotation interface that allows the user to mark the bounding box end points on the hand-drawing. In practice, it takes less than ten seconds to mark out the four end points for a limb.

Intuitively, the radii of the two faces,  $r_1$  and  $r_2$ , are determined by back-projecting the image plane bounding box ( $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ )

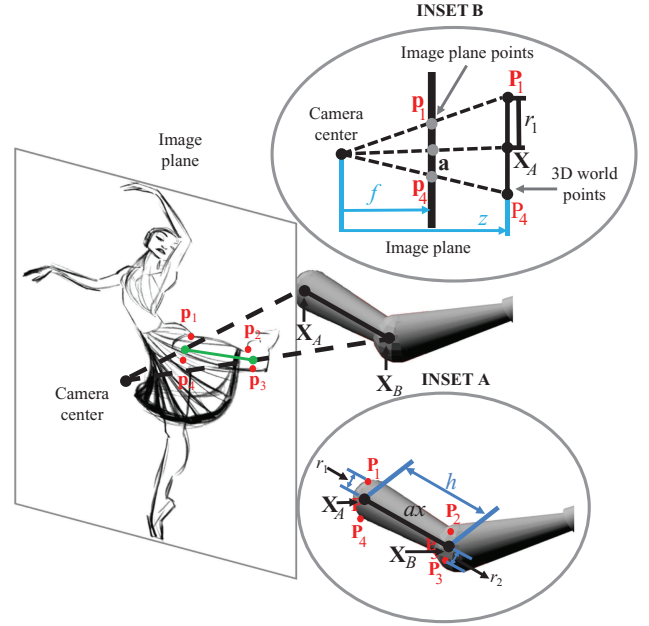


Fig. 6. Collision volumes: User-specified markers are back-projected to obtain the 3D marker positions  $\mathbf{X}_A$  and  $\mathbf{X}_B$ . Inset A: The cylindrical collision volume is characterized by its axis and height, and the radii of each face. Inset B: This inset describes how we compute the radius of one face of the cylinder. The image plane points  $\mathbf{p}_1, \mathbf{p}_4$  are back-projected to  $\mathbf{P}_1, \mathbf{P}_4$  such that the z-depth is the same as the z-depth for the marker A.

to the same z-depth as the 3D markers  $\mathbf{X}_A$  and  $\mathbf{X}_B$  (Figure 6, Inset B). Then, we spin the back-projected quadrilateral about the axis  $\vec{a}\vec{x}$  and the volume of revolution is the tapered cylinder proxy for the limb. Formally, for each point  $\mathbf{p}_q$  ( $q = 1, 2, 3, 4$ ), let  $\mathbf{P}_q$  denote their back-projected 3D positions. The input-match term  $e_a$  is redefined so that the 3D positions  $\mathbf{P}_q$  align with the image plane bounding box,

$$e_a = \|\mathbf{p}_q - \mathbf{p}_q^{proj}\|, \quad (7)$$

where  $\mathbf{p}_q^{proj} \cong \mathbf{M}\mathbf{P}_q$ . The error term  $e_m$  is redefined so that the z-depth for  $\mathbf{P}_q$  is the same as the z-depth for the markers (illustrated in Figure 6, Inset B),

$$e_m = \|\mathbf{m}_3^T \mathbf{P}_q - \mathbf{m}_3^T \mathbf{X}_A\| \quad \text{for } q = 1 \text{ and } 4, \quad (8)$$

$$e_m = \|\mathbf{m}_3^T \mathbf{P}_q - \mathbf{m}_3^T \mathbf{X}_B\| \quad \text{for } q = 2 \text{ and } 3. \quad (9)$$

We do not have a regularization term because the 3D positions  $\mathbf{X}_A$  and  $\mathbf{X}_B$  are already temporally smooth. The weighted sum of the error terms is linearized and minimized as detailed in the Appendix. The radii for the faces of the tapered cylinder are computed as

$$r_1 = \frac{\sqrt{\|\mathbf{P}_1 - \mathbf{P}_4\|}}{2}, \quad r_2 = \frac{\sqrt{\|\mathbf{P}_2 - \mathbf{P}_3\|}}{2}. \quad (10)$$

The sphere for a joint has the same radius as the corresponding tapered cylinder for the limb. Figure 7 shows the collision volumes overlaid with the hand-drawings on the left, and an alternate view-point of the same pose on the right of each part. The 3D proxy does not exactly conform to the artist's lines, but is sufficient to create believable interactions with dynamic simulations like cloth. Because the collision volumes can change size and shape independent of each other, this 3D proxy is well-suited to track the body of



Fig. 7. Collision volumes: The hand-drawings are shown overlaid with the collision volumes. The collision volumes match the drawings approximately, but need not exactly conform to the artist’s lines. We also show the collision volumes from an alternate viewpoint.

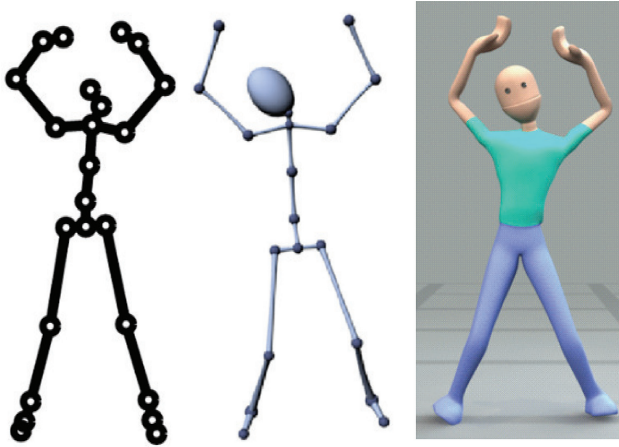


Fig. 8. Modified motion capture pose, 3D markers, hierarchical skeletal model.

a hand-animated character, even when the artist changes its size and shape (for example, if a dancer is drawn taller in frame 10 relative to frame 1 to create the illusion of reaching up).

**3.4.3 Skinned, Lit, and Rendered Model.** The third level of detail for the 3D proxy of the hand-drawn character is a skeletal model, which is controlled via a joint hierarchy, and can be skinned, lit, and rendered (Figure 8, right). When creating this proxy, the goal is to create 3D motion for the skeleton based on the artist’s hand-drawn animation. Because this 3D proxy can be rendered from any possible viewpoint, the error terms in our minimization are reformulated to maintain naturalness of motion and preserve ground contacts while transferring the style of the hand animation onto the skeleton.

The user-specified virtual markers on the hand drawings,  $\tilde{\mathbf{x}}_{ij}$ , contain the squash and stretch used by the artist to convey emotion. Because there is no precise mathematical model for how artists extend and compress the limbs of a character, the intentional change in limb lengths is indistinguishable from imprecision in the drawing and noise introduced during annotation. However, the limb lengths for the skeletal model are fixed. As a result, computing joint angle values directly from the user-specified virtual markers leads to large errors in joint angles.

We reformulate the input-match error term to filter out some of this noise. Recall that for each hand-drawn pose  $\tilde{\mathbf{x}}_i$ , the corre-

sponding pose in the motion prior segment,  $\tilde{\mathbf{X}}_i$ , was preprocessed so that the pose descriptors matched (Section 3.2). The upper body of the motion prior pose was modified while the lower body was left unchanged to preserve ground contacts. The input-match term for the skeletal model proxy aligns the 3D proxy to the modified pose  $\mathbf{x}_i^m$ , instead of the original hand-drawn pose  $\tilde{\mathbf{x}}_i$  (Figure 8, left and middle). Thus,

$$e_a^i = \sum_{j=1}^N \|\mathbf{R}\mathbf{X}_{ij}^w - \mathbf{x}_{ij}^m\|, \quad (11)$$

where  $\mathbf{R}$  was the orthographic approximation to the artistic camera. The motion prior pose  $\tilde{\mathbf{X}}_i$  is a reasonable prior for the desired 3D marker positions  $\mathbf{X}_i^w$ . Thus, the motion prior error term,  $e_m^i$ , is simply

$$e_m^i = \sum_{j=1}^N \|\mathbf{X}_{ij}^w - \tilde{\mathbf{X}}_{ij}\|. \quad (12)$$

Because the individual limbs of the 3D skeletal model are not squashable or stretchable, we additionally enforce constant limb lengths. Let the 3D positions of the two end-point virtual markers for a given limb be  $\mathbf{X}_{ij_1} = (x_{ij_1}, y_{ij_1}, z_{ij_1})$  and  $\mathbf{X}_{ij_2} = (x_{ij_2}, y_{ij_2}, z_{ij_2})$ . Then, the computed squared length of the limb,  $l_{ij}^2$ , is

$$l_{ij}^2 = (x_{ij_1} - x_{ij_2})^2 + (y_{ij_1} - y_{ij_2})^2 + (z_{ij_1} - z_{ij_2})^2. \quad (13)$$

This computed length must be equal to the actual skeletal length  $L_j$  of the same limb, computed from the motion prior data. Mathematically,

$$e_r^i = \sum_{j=1}^N \|l_{ij}^2 - L_j^2\|, \quad (14)$$

We can linearize Eq. (14) using a Taylor series expansion around the corresponding motion prior pose and stack the length equations for each limb to yield

$$e_r^i = \|\mathbf{A}_{limb}^i \mathbf{X}_i^w - \mathbf{b}_{limb}^i\|, \quad (15)$$

where  $\mathbf{A}_{limb}^i$  and  $\mathbf{b}_{limb}^i$  are functions of the motion prior pose  $\tilde{\mathbf{X}}_i$  and  $L$ .

The three error terms are stacked together as a linear system of equations

$$\mathbf{W} \begin{bmatrix} \mathbf{R} \\ \mathbf{A}_{limb}^i \\ \mathbf{I} \end{bmatrix} \mathbf{X}_i^w = \begin{bmatrix} \tilde{\mathbf{x}}_i^m \\ \mathbf{b}_{limb}^i \\ \tilde{\mathbf{X}}_{ij} \end{bmatrix}, \quad (16)$$

$$\mathbf{W}\mathbf{A}_{full}^i \mathbf{X}_i^w = \mathbf{b}_{full}^i. \quad (17)$$

where  $\mathbf{W}$  contains the weights of the various error terms.

Typically, the number of markers for each pose is  $N = 24$ , which makes  $\mathbf{X}_i^w$  a vector of length 72. The work of Safonova and colleagues [2004] shows that most dynamic human motions can be described by a low-dimensional PCA (Principal Components Analysis) subspace. Building on this result, we look for the solution to Eq. (17) in a low-dimensional subspace of an activity-specific motion capture database. Let  $\mu$  denote the mean of the motion data and  $\mathbf{v}_k$  denote the basis vectors or the principal components obtained through PCA. Further, let  $\mathbf{X}_i^b$  be the coordinates in PCA space, and  $\mathbf{V}$  be a  $3N \times P$  matrix of basis vectors. We have

$$\mathbf{X}_i^w = \mu + \sum_{k=1}^P x_{ik}^b \mathbf{v}_k \quad (18)$$

$$= \mu + \mathbf{V}\mathbf{X}_i^b, \quad (19)$$

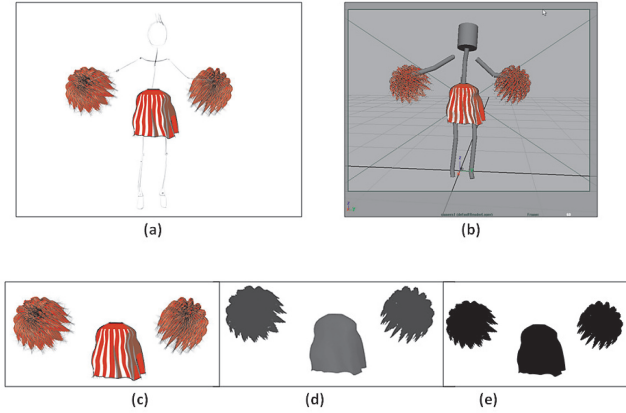


Fig. 9. (a) The final composited frame with the hand-drawn character and the rendered 3D elements. (b) Collision volumes imported into the 3D animation package. (c) Rendered scene elements. (d) Z-depth for the rendered elements. (e) Occlusion map for the rendered elements.

where  $\mathbf{X}_i^b$  is a vector comprising the coordinates for each of the  $P$  basis vectors. The weighted least squares system in Eq. (17) can then be written as

$$\mathbf{WA}_{full}^i (\mathbf{VX}_i^b + \mu) = \mathbf{b}_{full}^i, \quad (20)$$

$$\mathbf{WA}_{full}^i \mathbf{VX}_i^b = \mathbf{b}_{full}^i - \mathbf{WA}_{full}^i \mu, \quad (21)$$

$$\mathbf{AX}_i^b = \mathbf{b}. \quad (22)$$

We can find the least squares solution to Eq. (22) and reproject  $\mathbf{X}_i^b$  to get the 3D marker positions  $\mathbf{X}_{ij}^w$ . As this system is linear, the solution is the global minimum, is numerically stable, and can be found in closed form.

In a hierarchical skeletal model, every limb is described by three joint angles (roll, pitch, and yaw) relative to its parent limb. We convert the marker positions  $\mathbf{X}_i^w$  into joint angles. The root joint for our character model is the pelvis, therefore, we start by recovering the rotation of the pelvis with respect to the world coordinate frame and work our way through each hierarchical chain to generate the full pose [Jain et al. 2009]. The joint angles for our skeletal model describe the rotation of the limb segment in the  $xyz$  ordering; when we convert this description to the  $zyx$  ordering,  $\theta_x$  and  $\theta_y$  are functions of 3D marker positions (roll and pitch), and  $\theta_z$  is the “yaw” angle, which cannot be computed from marker positions. We find the rotation of the corresponding limb segment in the motion prior, and simply use that  $\theta_z$  to complete the generated 3D pose (Figure 8).

### 3.5 Interaction between Proxy and Scene Elements

The 3D proxy model can be imported into any commercially available modeling and animation software package. As a 3D scene element, the proxy can interact with other objects in the scene. For example, the hierarchical joint model could be imported into a Maya scene, and skinned. We can then put a dynamically simulated skirt on the 3D model, relight the scene, and move the camera around, as in Figure 18.

In Figure 9, the collision volumes for the jumping jacks character have been imported into a Maya scene. An artist has created pompoms and a skirt. The Maya dynamics engine [Stam 2009] is used to physically simulate the motion of the pompoms and the skirt, and their interaction with the 3D collision volumes of the hand-

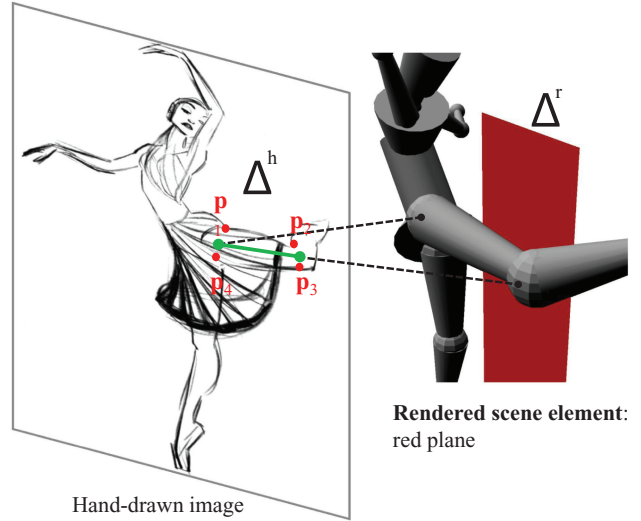


Fig. 10. Depth ordering: Our method generates an alpha map for the hand-drawn image that maintains depth ordering between the hand-drawn pixels and the rendered 3D scene elements. The depth map for the 3D elements are rendered. The depth map for the hand-drawn character is computed from known marker depth values.

drawn character. Maya is also used to render the “beauty” pass,  $\Upsilon_i^r$ , which contains the scene elements with texture and lighting (not the 3D polygons comprising the proxy), the depth map  $\Delta_i^r$ , and the occlusion map  $\eta_i^r$  (Figure 9(c)–(e)).

We composite the rendered skirt and pompoms with the hand-animating to obtain simulated secondary motion for the jumping jacks character (the same procedure is used for all the animation sequences). The depth map  $\Delta_i^h$  for the hand-drawn image is computed by linearly interpolating known depths. For the skinned characters, the pixels belonging to a given limb are obtained by color segmentation (color coding done as part of user input in Section 3.1). For stick figures, we segment out the dark pixels by thresholding inside an oriented window along the limb  $v$  ( $v = 1, 2, \dots, V$ ).

The z-depth values for the pixels corresponding to the  $N$  virtual markers are known and can be interpolated to generate  $\Delta_i^h$ . For simplicity, we will drop the indices and denote the known depths as  $\bar{\mathbf{x}}$ . Let  $\mathbf{l}$  denote the line joining the end-point markers for limb  $v$ , whose image positions are  $\bar{\mathbf{x}}_a = (a_x, a_y)$  and  $\bar{\mathbf{x}}_b = (b_x, b_y)$ . Then,  $\mathbf{l}$  can be computed

$$\mathbf{l} = \frac{\bar{\mathbf{x}}_b - \bar{\mathbf{x}}_a}{\|\bar{\mathbf{x}}_b - \bar{\mathbf{x}}_a\|}. \quad (23)$$

Every pixel  $\bar{\mathbf{p}} = (\bar{p}_x, \bar{p}_y)$  belonging to the limb is assigned the same depth as the point  $\mathbf{p}$  closest to it on  $\mathbf{l}$ . We perform this interpolation for every limb in turn to obtain the depth  $\Delta_i^h$ , and then scale it to match the units of  $\Delta_i^r$  (Figure 10).

The occlusion map  $\eta_i^h$  for the hand-drawn frame is

$$\eta_i^h = \begin{cases} 1, & \text{in the interior of the hand-drawn figure,} \\ 1, & \text{on the artist's lines,} \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

The alpha matte  $\alpha_i$  for the hand-drawn frame  $\Upsilon_i^h$  is defined as the inverse of the gray-scale value,  $\alpha_i = (255 - \Upsilon_i^h)/255$ . This alpha



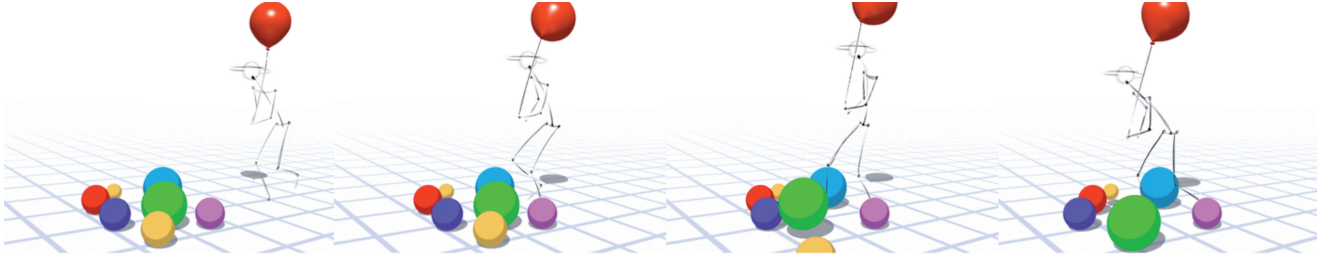


Fig. 11. Stylized walk across the screen: The dynamics of the balloon, its string, and the colored balls are driven by the motion of the hand-drawn character.



Fig. 12. Stylized walk across the screen: In the top row, snow deforms as the character steps through it. The deformation is generated by cylinder proxies for the feet of the character. In the bottom row, simulated rain bounces off a 3D umbrella attached to the wrist. The umbrella is attached to the hand-drawn character via a single-point proxy for the wrist.

matte is modified to incorporate depth ordering. For all pixels  $p$  where  $\eta_i^h(p) = 1$ ,

$$\alpha_{\text{new}_i} = \begin{cases} \alpha_i, & \text{if } \Delta_i^h < \Delta_i^r, \\ 0, & \text{otherwise.} \end{cases} \quad (25)$$

The final composited image  $I_{\text{final}_i}$  is computed

$$I_{\text{final}_i} = \alpha_{\text{new}_i} \Upsilon_i^h + (1 - \alpha_{\text{new}_i}) \Upsilon_i^r. \quad (26)$$

Because the final image is obtained by compositing the rendered 3D elements with the original hand drawing (not its 3D proxy), the visibility of the 3D elements is correct as long as the depth map  $\Delta_i^h$  is generated with the correct depth ordering.

## 4. RESULTS

We present results on a variety of hand-animated characters: a ballet dancer, a goofy character doing jumping jacks, a character doing a stylized walk across the screen, a “happy” flower, a little girl twirling, and a character ducking low. The hand animations include stick figures and fully fleshed characters, as well as stationary cameras and tracking cameras.

In Figure 11, the hand-drawn character “holds” a simulated 3D balloon and “kicks” the balls on the ground. The 3D balloon is attached to the single-point 3D proxy for the hand-drawn character,

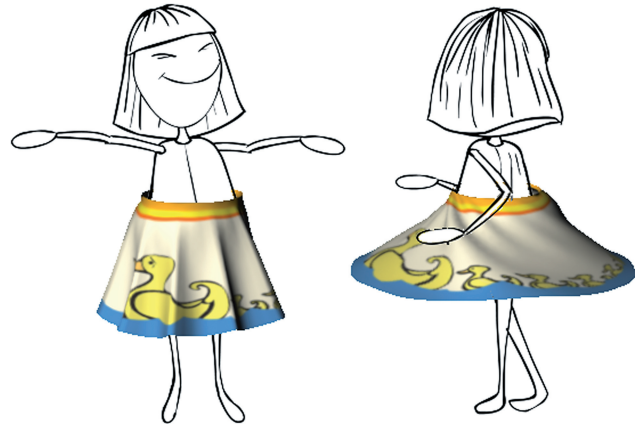


Fig. 13. Twirling girl: Two frames from an animation of a little girl showing off her new skirt. The girl is hand-animated. The skirt is a 3D cloth simulation driven by a 3D cylinder proxy for the girl’s body.

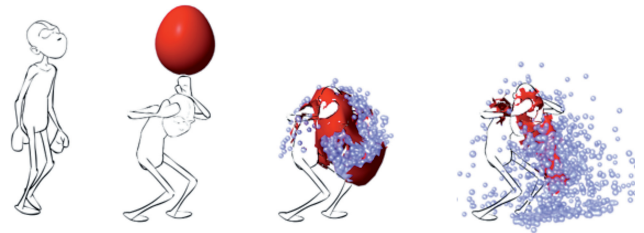


Fig. 14. Pedestrian: A simulated water balloon falls from the sky on a hand-drawn pedestrian. The balloon is a cloth simulation and is filled with water particles. Both the cloth and the water interact with the collision volumes for the hand-drawn character. The camera tracks the character as he walks and stops when he ducks.

that is, the 3D trajectory of the wrist marker. Cylinder proxies for the feet cause the balls on the floor to move. Figure 12 shows sample frames from the animated sequence of a character performing a stylized walk. The example in the top row shows the deformation of the snow, driven by a 3D cylinder proxy that approximates the character’s foot. In the bottom row, the umbrella is attached to the 3D marker on the wrist of the character. Simulated rain splashes and bounces off the 3D umbrella.

Figures 13, 14, and 15 show results with cloth simulations. In Figure 13, the 3D proxy for the little girl is the tapered cylinder model. The skirt is a cloth simulation driven by the 3D proxy, then



Fig. 15. Ballet dancer: Scarves are simulated as 3D cloth. They are attached to the wrists of the dancer via single-point proxies for the wrist, and interact with the 3D cylinder proxy for the body of the hand-drawn dancer, and thus, wrap around her waist.

rendered and composited onto the hand animation. Thus, the simulated skirt appears to be twirled by the hand-drawn little girl. Figure 14 shows a water balloon falling from the sky on a pedestrian. The pedestrian is a hand-animated character. The balloon is a simulated cloth object and the water inside it is comprised of particles. The balloon bursts when it comes in contact with the 3D proxy for the character and we see it snag on his body on subsequent frames. In Figure 15, two scarves are attached to the wrists of the dancer. The cloth simulation is driven by the three-dimensional trajectory of the wrist proxies, and interacts with the 3D polygons for the body of the ballerina.

Figure 16 demonstrates the advantage of leveraging 3D simulation to create secondary motion for a hand animation. Delicate effects like the strands of the pompoms, water splashing in a realistic way, and the bounce of a clothesline would all be time consuming to hand animate to a comparable degree of detail. The pompoms are attached to the character’s wrist, while all the other effects use the tapered cylinder 3D proxy for the hand-drawn character.

We also present examples where we have transferred the style of the hand-animation onto hierarchical joint models. The joint model proxies for the stylized walk, jumping jacks, and “happy” flower sequences are compared with the corresponding motion prior poses in Figure 17; note how the arms in the 3D proxy pose match the hand-drawn character, even though the motion prior pose is

different. In Figure 18, we see the 3D ballerina from two different camera viewpoints.

In all the results presented, user effort can be divided into three parts.

- Cleanup/ink-and-paint stage: Marking out dots or user-specified virtual markers (1 minute per frame), marking bounding boxes (3–4 minutes per frame), color segmentation of body parts (7–10 minutes per frame using a stylus and tablet, and a standard brush-based paint program).
- Selecting a motion capture segment: 20–30 minutes.
- Creating 3D simulation in Maya: 2–10 hours (2 hours for the simple rigid bodies, 10 hours for the cloth). The tuning time is dependent on the user’s familiarity with the tool and is equivalent to the tuning required to add any visual effect to a 3D animation. Maya can be replaced by any other simulation engine.

## 5. EVALUATION

In our method, the user provides the motion prior by selecting a suitable motion capture segment from a database. This prior influences different error terms in the minimization depending on the required 3D proxy. We examine the dependence of our method on the motion

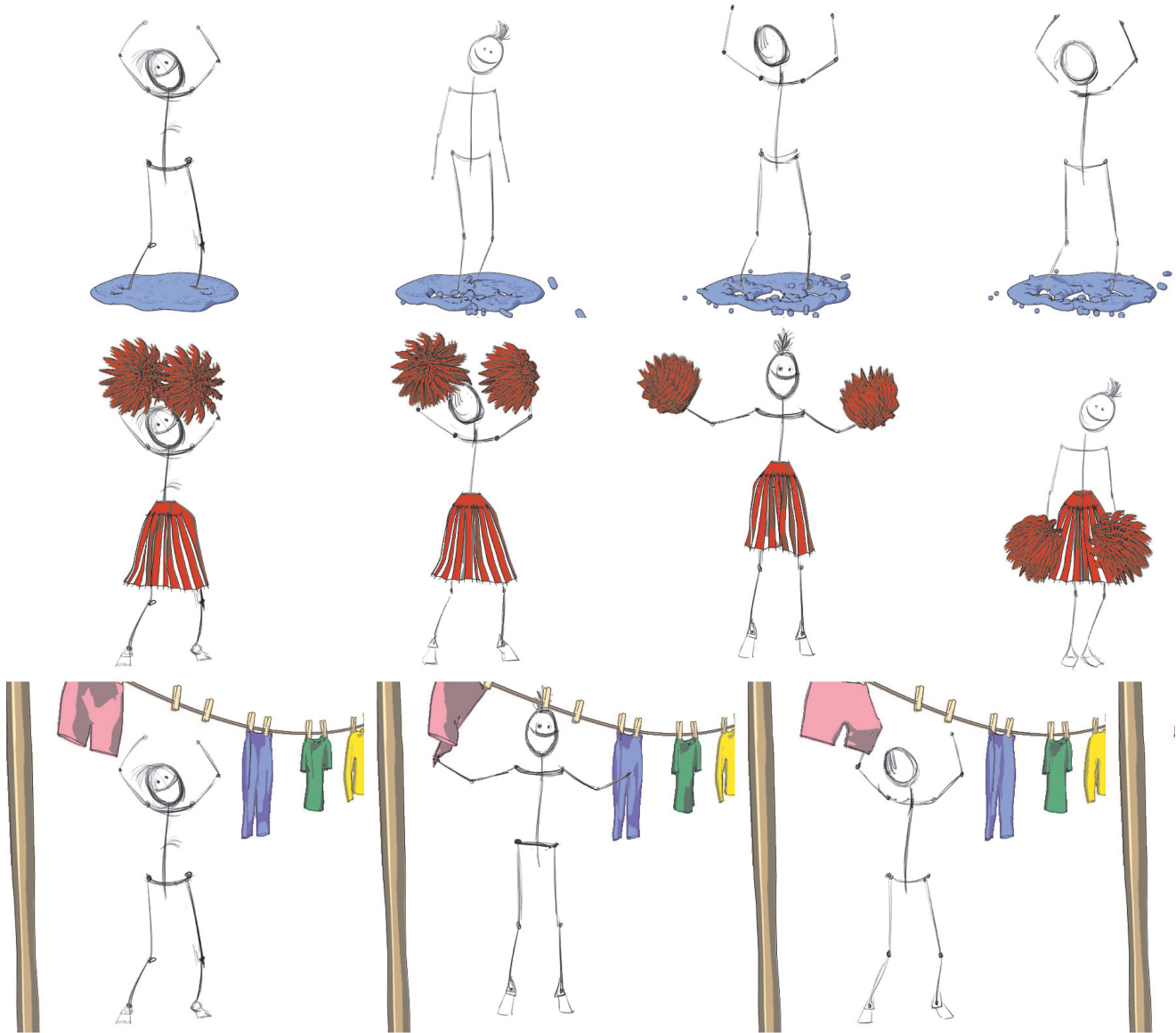


Fig. 16. Goofy character doing jumping jacks: Water splashes in response to the feet (top row); the pom-poms deform in a physically realistic way, and the skirt billows around the character’s legs (middle row); and the character’s hand interacts with clothes on the clothesline (bottom row).

prior through a synthetic example: a motion capture walk sequence (normal walk in a straight line) projected to 2D.

### 5.1 Motion Capture Data Provides Only z-Depth

Here, we compute the error in the 3D positions of virtual markers when the 2D markers of the synthetic example are back-projected under a known projection operator, and the z-depth is provided by different motion capture segments. Essentially, the motion prior  $\hat{\mathbf{X}}$  in Eq. (4) is computed from five motion capture segments: a walk sequence from a different actor, a run, a broad jump, a side shuffle, and a walk along a curve. Error is defined as the difference in z-depth value from ground truth, averaged over all  $N$  markers. Figure 19 illustrates that broad jump and run have similar numerical error

in depth; the depth ordering for the limbs is similar for these actions when viewed in profile. Figure 20 shows sample frames when the z-depth is provided by a “happy walk” motion capture segment. Even though the motion capture walk is stylistically different (arms swing more, legs step out further), the result matches the ground truth quite closely. Figure 21 shows sample frames when the z-depth is provided by a “curved walk”. Because the actor curves towards the camera, the changing depth information causes our result to curve towards the camera as well.

This evaluation shows that a motion capture segment that works well provides a reasonable prior for both root motion and relative depths of different limbs. In the case of the curved walk, the large error in z-depth is mostly because the motion of the root was different from ground truth. In the case of the side shuffle, root motion was

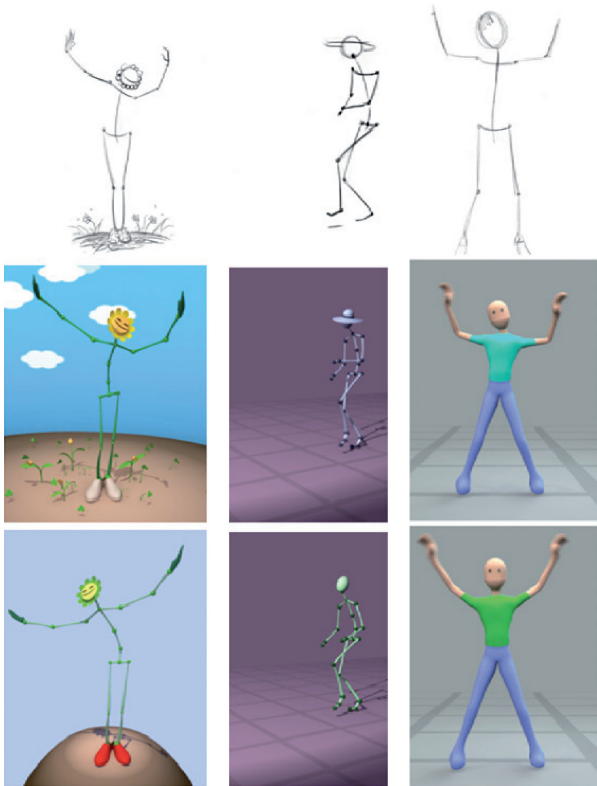


Fig. 17. For each character, the top row is the hand-drawn animation, the middle row shows our result, and the bottom row shows the corresponding motion capture poses. The best-matching motion capture segment in the database contains a similar sequence of actions, but not necessarily the same poses as the hand-animation. Even if the database were larger, the motion capture poses might never exactly match the hand-animated poses because of physical limits. For example, we found that the playful nature of jumping jacks animation (which was elegantly communicated by the artist by drawing a bobbing head) was quite difficult for a human actor to perform because it is hard to perform a vigorous action like jumping jacks while bobbing the head in sync. Modifying the motion capture pose via the pose descriptor allows our algorithm to create this playful head bob for the 3D proxy.



Fig. 18. The hand-drawn frame with the corresponding renderings from two views. The skirt is a dynamically simulated element. The lighting can be changed easily.

quite similar to the ground truth (straight line, maintaining almost the same depth from the camera for the entire duration). However, the relative depths of the limbs were completely different. The happy walk balanced both these priors.

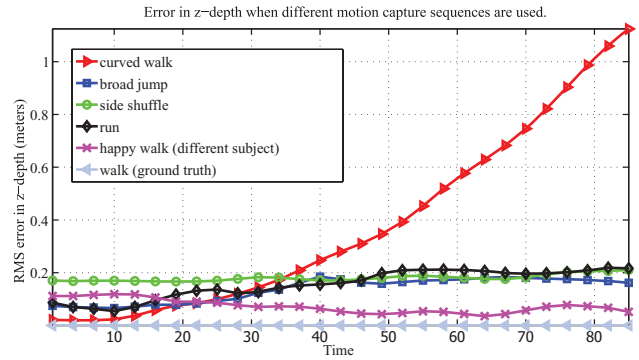


Fig. 19. Different motion capture segments affect the error in z-depth. The normal walk is the ground truth. The z-depth error for the curved walk increases as the motion capture poses veer towards the camera. The lowest error is seen in a happy walk sequence captured on a different actor and this motion could be used as a driving signal. Run and side shuffle have smaller errors than the curved walk, but for detailed interaction, these motions would probably also not provide sufficiently accurate z-depth values.

## 5.2 Motion Capture Data Influences Input-Match Term

In Section 3.4.3, we reformulate the input-match term  $e_a$  to align the 3D proxy to the modified motion capture pose, rather than the original hand-drawn pose. The modification is intended to transfer the style of hand-animation onto the 3D proxy, while filtering out noise. Here, we evaluate the question: was the modification successful, or are we simply playing back the motion prior? Numerically, the modification should cause the 3D proxy to be a better match to the artist-drawn animation than the motion prior.

The input is a motion capture walk (“normal walk”) projected to 2D. The motion prior is chosen to be a stylistically different walk (happy walk) captured on a different subject. We compare the distance between markers on the ground truth with markers on the motion prior and the 3D proxy. The Root Mean Squared (RMS) error for each virtual marker is shown in Figure 22. The ticks on the  $x$ -axis represent the virtual markers. Ticks 1 through 42 are markers on the “upper body”, and Ticks 43 through 72 represent markers on the “lower body”. The RMS error between the ground truth marker positions and the 3D marker positions of the “happy walk” motion prior, averaged over a 100 frame segment, are shown in red. Because the limb segments belonging to the upper body are modified to match the pose descriptor of the synthetic input markers, we can see that the error for these markers is reduced in the 3D proxy (blue), while the error for the lower body markers is unchanged, as expected.

## 6. DISCUSSION

In this work we have shown that connecting hand-drawn animation with 3D computer animation offers many advantages; simulating physically-driven clothes for a hand-drawn character, for example, or empowering traditionally trained animators to contribute to a 3D character’s performance. This connection is made possible by generating a 3D representation, or proxy, for the 2D hand-drawn character, and allowing the proxy to interact with the other scene elements in the 3D virtual world. Our key observation is that we should create different levels of detail in the 3D proxy for different

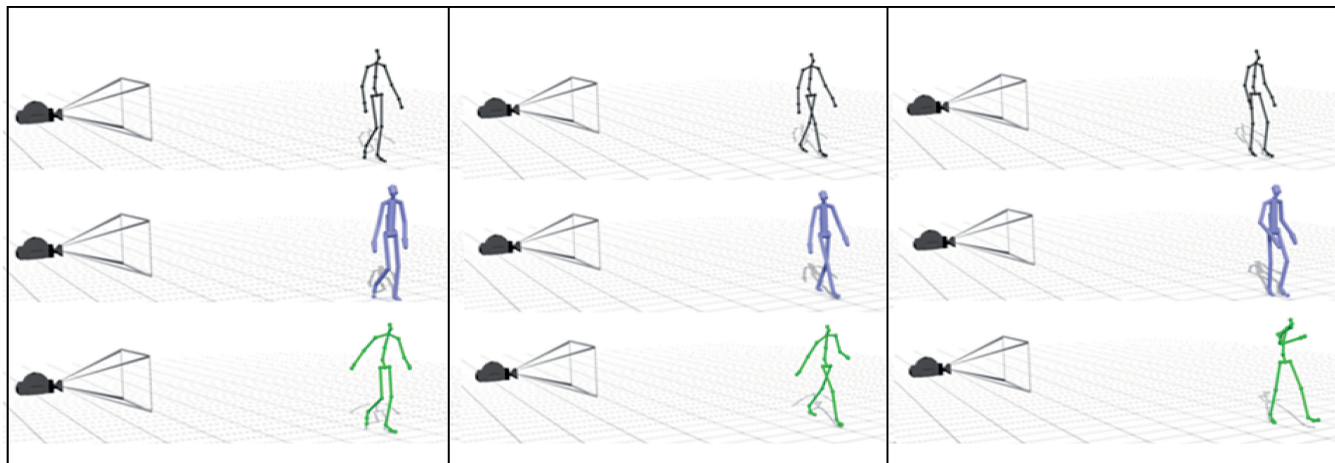


Fig. 20. We compare the ground-truth walk (black, top row) with our result (purple, middle row), and the motion capture segment which provided z-depth information (green, bottom row). This motion capture segment is a “happy walk”, where the actor walked in the same direction but swung his arms and legs higher to communicate happiness. Even though the 3D marker positions for the motion capture poses are quite different, the result is close to the ground-truth pose because the motion capture poses provide only depth.

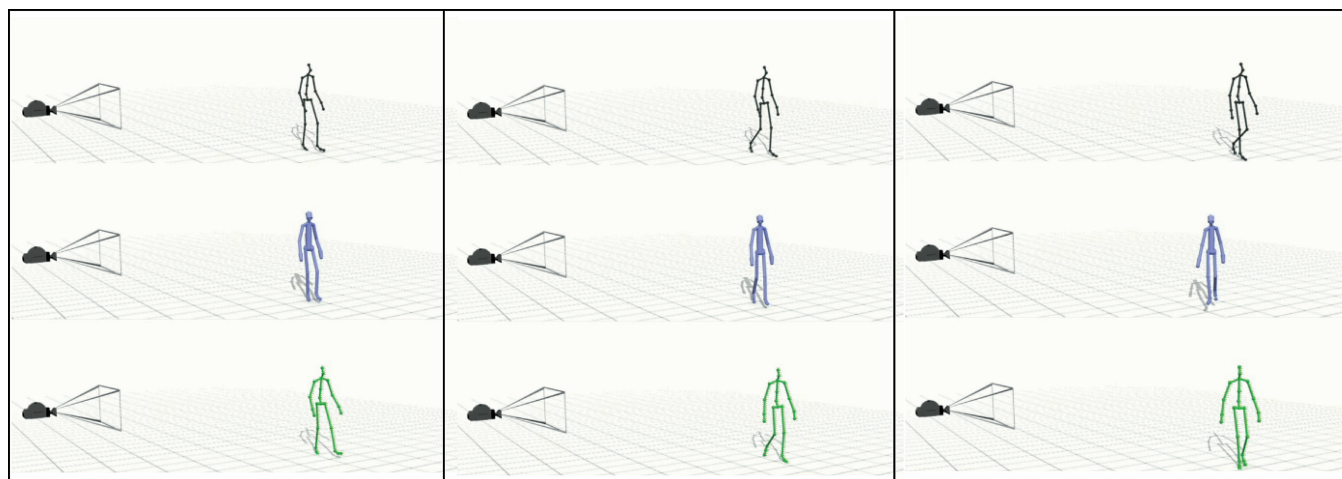


Fig. 21. We compare the ground-truth walk (black, top row) against our result (purple, middle row), when z-depth information is provided by a “curved walk” motion capture segment (green, bottom row). Because the motion capture poses curve towards the camera, the z-depth values are not the same as the z-depth values for the ground-truth poses. We can see that our result also curves towards the camera because the result poses have the same z-depths as the motion capture poses.

levels of interaction between the hand-drawn character and the 3D scene elements.

We reformulate three basic error terms for each level of detail and show that they may be linearized and thus, minimized in closed form. The first error term, the input-match term, causes the 3D proxy to follow the hand-animation. A limitation of our method is that the skeletal model 3D proxy will follow the style of the hand-drawn animation in the upper body movement, but will move like the motion capture segment in the lower body. For example, if the hand-drawn ballerina lifts her leg higher than the motion captured actor in *attitude derrière* (Figure 23, top row), then the 3D proxy ballerina will not match the hand-drawn leg pose exactly. This mismatch occurs because the skeletal model 3D proxy is generated by back-projecting the modified motion capture poses. The lower body limbs

are not modified with the pose descriptor because rotating the leg and feet segments would result in loss of ground contact. The hand-drawn character is proportioned differently than the motion capture model, and its body proportions also vary across the frames of the hand-drawn sequence. Therefore, the lost ground contact cannot trivially be corrected by a global translation of the pelvis. This limitation of the pose descriptor can be alleviated in part by having a large motion capture database from which to select the motion priors. Alternately, we could plant the feet explicitly, and solve for a skeleton via nonlinear optimization. As this is susceptible to local minima solutions, future work would be to linearize such constraints.

The motion prior error term provides a data-driven prior for inferring the z-depth. Our method is thus limited to hand-animations

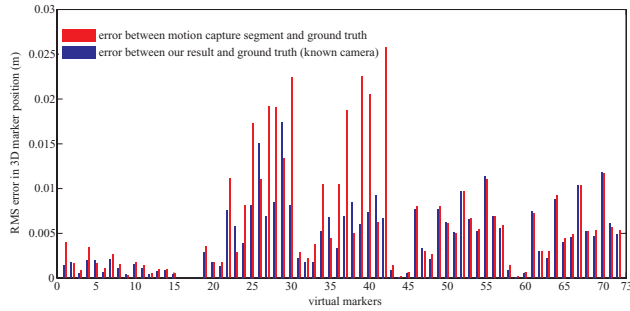


Fig. 22. Error in 3D world positions for virtual markers when motion capture data is modified via the pose descriptor.

of human-like characters, or nonhumanoid characters for which 3D animated motion priors were available by keyframing. Though our method uses a contiguous motion capture sequence, techniques such as time warping, retargeting, motion blending, and motion re-sequencing would allow the generation of better motion priors for a given hand-animation (using, for example, Gleicher [1998], Lee et al. [2002], Baran et al. [2009], Ikemoto et al. [2009], Zhao and Safonova [2009]).

An advantage of drawing on motion capture data as a prior is that it keeps the solution well-behaved. In Section 3.4.3, we had minimized the linear system in a low-dimensional subspace and then projected it to full space to obtain the 3D world positions for the virtual markers. We investigated how the solution would change when the number of principal components  $P$  was increased. More dimensions allow us to capture finer details, as can be seen in the elbow and wrist angle in Figure 23. The motion prior keeps the solution well-behaved even in the full-dimensional space. Also, our method does not include nonpenetration constraints as part of the error terms in the minimization. As a result, it is possible for the limbs of the 3D proxy to intersect. Using well-behaved motion capture data as a motion prior alleviates this problem, though it does not guarantee nonpenetration. Simulating tight-fitting clothes, such as a shirt with sleeves, might be difficult as a result of this limitation.

In our method, a user specifies the camera moves implicitly, by selecting a motion capture segment. Thus, if the database contains only an overground walk cycle, and the hand-animation contains the character walking in place (on a treadmill), our method will assume that the character walks overground with a tracking camera. The water balloon result in Figure 14 is an example that stress-tests the assumption that the best-matching motion capture segment will be sufficient to estimate the camera imagined by the artist. In the animated movie (included as supplementary material), we see that the estimated camera tracks the character as he walks, but moves very slightly instead of remaining completely stationary when the character ducks. This movement happens because it is not reasonable to find, or capture, a motion capture segment where the actor has ducked with the arms and legs bending exactly in the same order as in the hand-animation; for example, the actor might brace her arms over her head before she bends at the knees. As a result, our camera estimation routine incorrectly infers that the camera should move to account for this mismatch. Future work could incorporate information about camera moves from the shot exposure sheet or add annotation about the world coordinate frame in the process of animating. A possible annotation could be a cube drawn on the ground plane in every frame, thus providing a world reference.

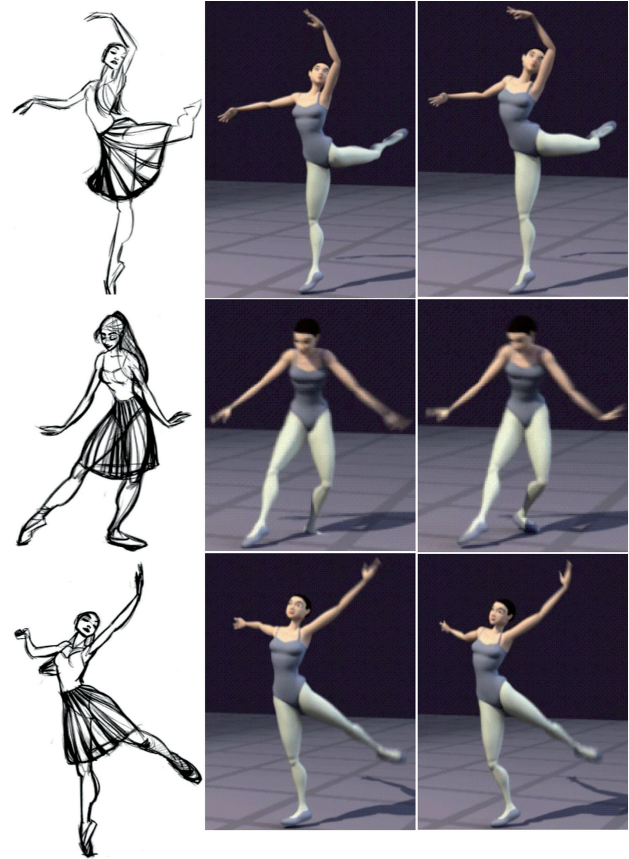


Fig. 23. Projection to a low-dimensional subspace: We varied the number of principal components  $P$  used to represent the 3D pose. Left: hand-drawn frames. Center:  $P = 20$ . Right:  $P = 72$  (all dimensions). More dimensions allow for finer details, as seen in the elbow and wrist.

As part of preprocessing, in Section 3.2, we proposed a pose descriptor to modify motion capture data. The pose descriptor can be thought of as a shape descriptor that retargets the pose of the hand-drawn figure onto a skeleton proportioned to match the motion capture data. This step allows us to transfer the pose from a hand-drawn ballerina, who may be statuesque, onto a “normal” female actor, or the pose of a goofy character with long arms onto a “normal” male actor, and then use human motion capture data as a motion prior. This pose descriptor preserves in-plane joint angles in its current formulation. A useful future work direction would be to extend it to preserve contacts or other metrics that might better capture the essence of the hand-drawn animation.

Though all our results have used hand-animations as input, we could apply the same concepts to 3D proxies for video sequences; for example, to add a scarf on a person walking outdoors on a windy day. We could draw on the literature in the computer vision and graphics communities on transferring lighting, shadows, and other visual cues to match the added 3D elements to the video elements. An interesting research question would be whether the addition of 3D elements can be made seamless to viewers. Even though we can match the rendering style of the added 3D elements (by toon-shading to match hand-animations, for example), the movement of the 3D elements would need to match the original video or animation as well. For example, physical parameters of the cloth simulation, like

gravity and friction, might need to be modified so that the pom-pom and skirt simulations are a better match to the bounciness of the hand-drawn jumping jacks without tangling or crumpling.

We have proposed three different levels of detail in a 3D proxy depends on its function. We could extend this line of thought to more levels of detail, for example, a mesh model that conforms to the artist's lines, thus allowing 3D texture-mapping and lighting techniques to be used for hand-drawn characters. Li and colleagues describe a method to deform a mesh model once the driving skeleton has been modified to match the hand-drawn animation [Li et al. 2003]. We could use the method of Section 3.4.1 to match the driving skeleton to markers on the hand-drawn animation, thus automating one of the user inputs in their method.

## APPENDIX

The 3D position of marker  $j$  in frame  $i$  is expressed in homogeneous coordinates,  $\mathbf{X}_{ij}^w = [X_{ij}^w, Y_{ij}^w, Z_{ij}^w, 1]^T$ . Its projection on the image plane via the projection operator  $\mathbf{M}_i$  is defined up to scale,

$$\mathbf{x}_{ij}^{proj} \cong \mathbf{M}_i \mathbf{X}_{ij}^w.$$

The input-match term  $e_a^{ij}$  is defined

$$e_a^{ij} = \|\tilde{\mathbf{x}}_{ij} - \mathbf{x}_{ij}^{proj}\|. \quad (27)$$

This error term can be linearized though a cross product [Hartley and Zisserman 2003]. Eq. (27) is equivalent to

$$\tilde{\mathbf{x}}_{ij} \times \mathbf{M}_i \mathbf{X}_{ij}^w = 0. \quad (28)$$

On rearranging the cross product as a matrix operation Eq. (28) is written as

$$\mathbf{C}\mathbf{M}_i \begin{bmatrix} X_{ij}^w \\ Y_{ij}^w \\ Z_{ij}^w \\ 1 \end{bmatrix} = 0, \quad (29)$$

where  $\mathbf{C} = \begin{bmatrix} 0 & -1 & \tilde{y}_{ij} \\ 1 & 0 & -\tilde{x}_{ij} \\ -\tilde{y}_{ij} & \tilde{x}_{ij} & 0 \end{bmatrix}$ , and  $\mathbf{M} = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix}$ , are known matrices.

Intuitively, Eq. (29) constrains  $\mathbf{X}_{ij}^w$  to lie on a back-projected ray starting at the camera center, going through the image plane point  $\tilde{\mathbf{x}}_{ij}$ .

We stack Eqs. (29), (4), and (6) for each frame, and denote the combined linear system  $\mathbf{A}_{ij}$ ,

$$\mathbf{A}_{ij} \mathbf{X}_{ij}^w = \mathbf{b}_{ij}. \quad (30)$$

The smoothness term in Eq. (5) is incorporated into a large sparse matrix as follows:

$$\mathbf{W} \begin{bmatrix} \mathbf{A}_{11} & 0 & \dots & \dots \\ 0 & \mathbf{A}_{21} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \mathbf{A}_{KN} \\ \mathbf{I} & -\mathbf{I} & 0 & \dots \\ \dots & \mathbf{I} & -\mathbf{I} & \dots \\ 0 & \dots & \dots & \dots \\ 0 & \dots & \mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{11}^w \\ \mathbf{X}_{21}^w \\ \dots \\ \mathbf{X}_{KN}^w \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{11} \\ \mathbf{b}_{21} \\ \dots \\ \mathbf{b}_{KN} \\ \mathbf{0} \\ \dots \\ \mathbf{0} \end{bmatrix}, \quad (31)$$

$$\mathbf{W}\mathbf{A}_{full} \mathbf{X}_{full}^w = \mathbf{b}_{full}, \quad (32)$$

where  $\mathbf{W}$  is the weight matrix that describes the relative weights between the geometric constraints and the smoothing terms. We solve for the least squares solution to Eq. (32).

## ACKNOWLEDGMENTS

We thank Glen Keane, Tom LaBaff, and Travis Blaise for providing the input animations, and Justin Macey for help with motion capture. Thanks also to Autodesk for their donation of the 3D animation and rendering package Maya.

## REFERENCES

- AGARWAL, A. AND TRIGGS, B. 2006. Recovering 3d human pose from monocular images. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 1, 44–58.
- BARAN, I., VLASIC, D., GRINSPUN, E., AND POPOVIĆ, J. 2009. Semantic deformation transfer. *ACM Trans. Graph.* 28, 3, 1–6.
- BOURDEV, L. AND MALIK, J. 2009. Poselets: Body part detectors trained using 3d human pose annotations. In *Proceedings of the IEEE International Conference on Computer Vision*.
- BREGLER, C. AND MALIK, J. 1998. Tracking people with twists and exponential maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- COOPER, D. 2002. 2D/3D Hybrid character animation on Spirit. In *Proceedings of the ACM SIGGRAPH '02 Conference Abstracts and Applications Conference*.
- CORRÊA, W. T., JENSEN, R. J., THAYER, C. E., AND FINKELSTEIN, A. 1998. Texture mapping for cel animation. In *Proceedings of the ACM SIGGRAPH '98*. 435–446.
- CULHANE, S. 1990. *Animation From Script to Screen*. St. Martin's Press, New York.
- DANIELS, E. 1999. Deep canvas in Disney's Tarzan. In *Proceedings of the ACM SIGGRAPH '99 Conference Abstracts and Applications*.
- DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIC, Z., AND SALESIN, D. H. 2003. A sketching interface for articulated figure animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 320–328.
- DEMIRDJIAN, D., KO, T., AND DARREL, T. 2003. Constraining human body tracking. In *Proceedings of the IEEE International Conference on Computer Vision*.
- ELGAMMAL, A. AND LEE, C. 2004. Inferring 3d body pose from silhouettes using activity manifold learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- ELLIS, D. 2003. Dynamic time warp (DTW) in Matlab. [www.ee.columbia.edu/~dpwe/resources/matlab/dtw/](http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/).
- FORSYTH, D. A., ARIKAN, O., IKEMOTO, L., O'BRIEN, J., AND RAMANAN, D. 2005. Computational studies of human motion: part 1, tracking and motion synthesis. *Found. Trends Comput. Graph. Vis.* 1, 2-3, 77–254.
- GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of the ACM SIGGRAPH '98*. 33–42.
- GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIC, Z. 2004. Implicit surface joint limits to constrain video-based motion capture. *ACM Trans. Graph.* 23, 3, 522–531.
- HARTLEY, R. AND ZISSERMAN, A. 2003. *Multiple View Geometry*, 2 ed. Cambridge University Press.
- HERDA, L., URTASUN, R., AND FUA, P. 2004. Implicit surface joint limits to constrain video-based motion capture. In *Proceedings of the European Conference on Computer Vision*. 405–418.
- HORNUNG, A., DEKKERS, E., AND KOBELT, L. 2007. Character animation from 2d pictures and 3d motion data. *ACM Trans. Graph.* 26, 1, 1–9.
- IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2009. Generalizing motion edits with gaussian processes. *ACM Trans. Graph.* 28, 1, 1–12.
- JAIN, E., SHEIKH, Y., AND HODGINS, J. K. 2009. Leveraging the talent of hand animators to create three-dimensional animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

- JAIN, E., SHEIKH, Y., MAHLER, M., AND HODGINS, J. 2010. Augmenting hand animation with three-dimensional secondary motion. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.
- JOHNSTON, O. AND THOMAS, F. 1995. *The Illusion of Life: Disney Animation*. Disney Editions.
- JOHNSTON, S. F. 2002. Lumo: Illumination for cel animation. In *Proceedings of the (NPAR'02) Symposium on Non-Photorealistic Animation and Rendering*, 45–52.
- LASSETER, J. 1994. Tricks to animating characters with a computer. In *ACM SIGGRAPH Course Notes*.
- LEE, H. J. AND CHEN, Z. 1985. Determination of 3d human body postures from a single view. *Comput. Vis. Graph. Image Process.* 30, 148–168.
- 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. *ACM Trans. Graph.* 21, 3, 491–500.
- LI, Y., GLEICHER, M., XU, Y.-Q., AND SHUM, H.-Y. 2003. Stylizing motion with drawings. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 309–319.
- MOESLUND, T. B. AND GRANUM, E. 2006. A survey of computer vision-based human motion capture. *Comput. Vis. Image Understand.* 81, 3, 231–268.
- PETROVIĆ, L., FUJITO, B., WILLIAMS, L., AND FINKELSTEIN, A. 2000. Shadows for cel animation. In *Proceedings of the ACM SIGGRAPH'00*, 511–516.
- RADEMACHER, P. 1999. View-dependent geometry. In *Proceedings of the ACM SIGGRAPH'99 Conference*, 439–446.
- RAMANAN, D., FORSYTH, D., AND ZISSERMAN, A. 2004. Tracking people by learning their appearance. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 1, 65–81.
- RIVERS, A., IGARASHI, T., AND DURAND, F. 2010. 2.5d cartoon models. *ACM Trans. Graph.* 29, 4.
- ROBERTSON, B. 1998. Mixed media. *Comput. Graph. World*, 32–35.
- ROSENHAHN, B., BROX, T., CREMERS, D., AND SEIDEL, H.-P. 2007a. Online smoothing for markerless motion capture. In *Proceedings of the DAGM Symposium on Pattern Recognition 4713*, 163–172.
- ROSENHAHN, B., BROX, T., CREMERS, D., AND SEIDEL, H.-P. 2008. Staying well grounded in markerless motion capture. In *Proceedings of the DAGM Symposium on Pattern Recognition 5096*, 385–395.
- ROSENHAHN, B., BROX, T., AND SEIDEL, H.-P. 2007b. Scaled motion dynamics for markerless motion capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.* 23, 3.
- SAKOE, H. AND CHIBA, S. 1990. Dynamic programming algorithm optimization for spoken word recognition. *Read. Speech Recogn.* 159–165.
- SIDENBLADH, H., BLACK, M., AND SIGAL, L. 2002. Implicit probabilistic models of human motion for synthesis and tracking. In *Proceedings of the European Conference on Computer Vision*.
- SIDENBLADH, H., BLACK, M. J., AND FLEET, D. J. 2000. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of the European Conference on Computer Vision*, 702–718.
- SMINCHISESCU, C., KANAUJIA, A., AND METAXAS, D. 2005. Discriminative density propagation for 3d human motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- SMINCHISESCU, C. AND TRIGGS, B. 2003. Estimating articulated human motion with covariance scaled sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- STAM, J. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *Proceedings of the IEEE International Conference on Computer-Aided Design and Computer Graphics*, 1–11.
- SYKORA, D., SEDLÁČEK, D., JINCHAO, S., DINGLIANA, J., AND COLLINS, S. 2010. Adding depth to cartoons using sparse depth (in)equalities. *Comput. Graph. Forum* 29, 2, 615–623.
- TAYLOR, C. J. 2000. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Understand.* 80, 349–363.
- URTASUN, R., FLEET, D. J., AND FUA, P. 2006. Temporal motion models for monocular and multiview 3d human body tracking. *Comput. Vis. Image Understand.* 104, 2, 157–177.
- WEI, X. AND CHAI, J. 2010. Videomocap: Modeling physically realistic human motion from monocular video sequences. *ACM Trans. Graph.* 29, 4, 1–10.
- WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. In *Proceedings of the ACM SIGGRAPH'97 Conference*, 243–250.
- WU, Y., HUA, G., AND YU, T. 2003. Tracking articulated body by dynamic markov network. In *Proceedings of the IEEE International Conference on Computer Vision*.
- ZHAO, L. AND SAFONOVA, A. 2009. Achieving good connectivity in motion graphs. *Graph. Models* 71, 4, 139–152.

Received May 2011; revised August 2011; accepted September 2011