Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2010)
M. Otaduy and Z. Popovic (Editors)

Augmenting Hand Animation with Three-dimensional Secondary Motion

Eakta Jain[1], Yaser Sheikh[1], Moshe Mahler[1], Jessica Hodgins[1,2]

[1]Carnegie Mellon University
[2]Disney Research, Pittsburgh

Abstract

Secondary motion, or the motion of objects in response to that of the primary character, is widely used to amplify the audience's response to the character's motion and to provide a connection to the environment. These three-dimensional (3D) effects are largely passive and tend to be time consuming to animate by hand, yet most are very effectively simulated in current animation software. In this paper, we present a technique for augmenting hand-drawn animation of human characters with 3D physical effects to create secondary motion. In particular, we create animations in which hand-drawn characters interact with cloth and clothing, dynamically simulated balls and particles, and a simple fluid simulation. The driving points or volumes for the secondary motion are tracked in two dimensions, reconstructed into three dimensions, and used to drive and collide with the simulated objects. Our technique employs user interaction that can be reasonably integrated into the traditional animation pipeline of drawing, cleanup, inbetweening, and coloring.

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

1. Introduction

Secondary motion is the motion of scene elements in response to the movement of the primary character and is often used to amplify the character's motion and personality via effects that appear to be driven by the motion. Examples of secondary motion include the swishing of a robe, or the sloshing of water in a pail. The animation of these effects is done after the character motion is finalized, either as a separate layer in hand-animation, or as a simulation in 3D animation. These elements do not, in general, have a distinct character or personality of their own—they are passive, not animate.

Passive effects such as cloth, fluids or particles have many degrees of freedom, are involved in frequent collisions, and have complex interactions with characters. These properties make them hard to animate by hand either on paper or via computer software. As a result, a great deal of research has focused on the simulation of these effects with impressive results in research prototypes that have subsequently been incorporated into commercial animation software. Three-dimensional (3D) or computer animation can easily take advantage of simulated effects as the character's location and pose is already known and can be used as a driving signal for the simulation. Hand, or two-dimensional (2D), animation cannot directly benefit from these techniques as the motion of the character is only determined on the image plane,

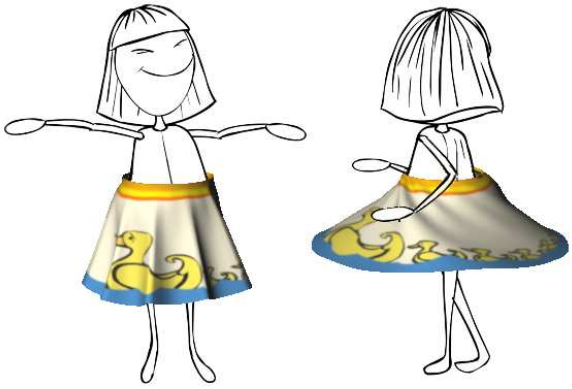

Figure 1: *Two frames from an animation of a little girl showing off her new skirt. The girl is hand-animated. The skirt is a 3D cloth simulation.*

© The Eurographics Association 2010.

DOI: 10.2312/SCA/SCA10/093-102

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

not in the full three dimensions required to drive and interact with a physical simulation.

In this paper, we propose a technique to add secondary motion onto a hand-animated character. Our goal is to preserve the animator's original lines, add secondary motion via existing simulation methods, and integrate well into the standard animation pipeline. Our approach makes three technical contributions. First, because the 3D reconstruction of a 2D signal is ambiguous, we resolve the 2D-3D ambiguity using z-depth information from motion capture data of similar behaviors to those performed by the animated character. Second, we create plausible collision volumes in 3D that interact with the desired physical simulation. Third, for the interaction to look believable, we composite the rendered effects with the hand-drawn frames while maintaining relative depth ordering.

We employ user interaction that can be reasonably integrated into the traditional animation workflow. We ask a user to annotate the joint locations with 'dots' for virtual markers and color-segment the various body parts. This user input fits well with the animation pipeline because each layer is already modified multiple times as it passes through the pipeline, and the required annotation is not a significant additional burden. In addition, we ask a user to select a similar motion capture segment from a large database. Once again, this user input requires little effort, and calls upon a resource that is easily available today. Other than this user input, the algorithm is automatic.

We envision that such a system could play a number of different roles in the creation of an animated sequence. If the physical simulation of secondary motion produces the desired effect, the results could be rendered and composited into the final production. The parameters of the simulation can be tuned to adjust the final effect by changing the material properties of the cloth or the viscosity of the water, for example. We have used this approach in the examples presented here. If the computed secondary motion is not exactly the desired effect or if rendering the effects in a style compatible with the hand animation is not possible, the rendered motion nonetheless could prove useful as a basis for rotoscoping with modifications as needed.

We present results on a number of hand-animated sequences, each of which has been augmented with effects created using the dynamics engine of the Maya software package and rendered using the Maya toon shader. The reconstruction of the driving signal is not specific to the details of either the simulation or the rendering engine used. We also evaluate our approach with a synthetic example to explore how similar the motion capture sequence must be to the hand animation.

2. Related Work

The computer graphics community has explored the idea of merging traditional animation with three-dimensional computer graphics (CG) animation. In this section, we discuss some examples of this work, and briefly survey related work in computer vision on reconstructing 3D human pose.

2.1. Computer Graphics

In taking 2D drawings to 3D, the ambiguity in the third dimension can either be resolved through user-intensive methods, or through domain knowledge. Sykora and colleagues [SSJ*10], employ user-specified depth inequalities to generate a 2.5D popup. Davis and colleagues [DAC*03] propose an interface that presents multiple 3D interpretations of a 2D hand-drawn pose to the user, sorted according to joint angle constraints and other heuristics—their algorithm does not attempt to resolve the depth ambiguity but instead allows the user to choose based on their intuition.

The recent work by Jain and colleagues [JSH09] employed domain knowledge to recreate a given hand animation in a 3D world, while maintaining its style. Their main contribution is a method to generate poses on a 3D hierarchical skeleton that are stylized, smooth and natural. Because they abstract away the hand-drawing during their reconstruction process, their final 3D animation consists of modified motion captured poses, which match the hand animation in style but do not exactly follow the animator's lines. For example, if the hand-drawn character is taller than the motion captured human, the markers of the reconstruction would not align with the hand-drawn limbs, although the joint angles would match.

Other works that incorporate artist-drawn input to create character animation include Li and colleagues [LGXS03], who modify the mesh parameters of motion captured animation to create exaggeration, and Bregler and colleagues [BLCD02], who semi-automatically animate a new character with the motion characteristics of a given 2D cartoon animation.

There has also been much research effort directed towards specific requirements of the traditional animation pipeline [CJTF98,PFWF00,Joh02,Anj01]. In particular, Petrovic and colleagues create ray-traced shadows on a 2D character by inflating it along the third dimension [PFWF00]. They propose an interface to make it easy for the user to specify the relative depths of scene elements.

Computer graphics techniques have been used to create background scenery, either in the form of 2D paintings manipulated to look three dimensional [WFH*97,Rob98], or as a 3D scene, as in *Tarzan*'s Deep Canvas [Tar99, Dan99]. In the movie *Spirit*, long shots were created as 3D scenes, while closeups were hand-animated [Coo02]. None of these works had the kind of physical interaction between the hand-drawn

© The Eurographics Association 2010.

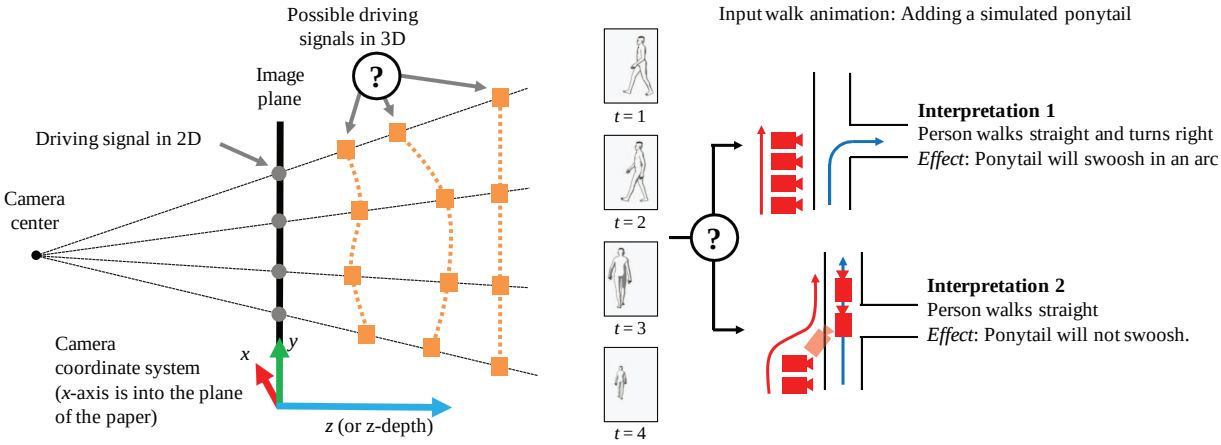

Figure 2: *(a) Depth ambiguity: multiple 3D trajectories can yield the same 2D projected path. (b) Composite motion ambiguity: the motion of the camera can not be disambiguated from the motion of the character if we are only given the image plane information.*

elements and the 3D elements needed to create secondary motion, for example 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.

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, [MG06] and [FAI*05]). Priors about the way humans are proportioned and how humans move have been used to make the estimation process tractable—these priors include limits on joint angles [ST03, HUF04], physical models of the body [RBS07], foot plants as a constraint [RBCS08], and known limb lengths [LC85, Tay00, WC09]. Sidenbladh and colleagues [SBF00] and Rosenhahn [RBCS07] used autoregressive models to apply smoothness constraints across a video sequence. Articulation constraints, i.e. ensuring that limbs must remain connected at joints, have also been used in a number of approaches [BM98, WHY03, DKD03]. Recently, dimensionality reduction methods, which rely on motion capture data to learn mappings, have become popular [SBS02, GMHP04, UFF06]. In contrast to these generative approaches, a number of discriminative approaches have also been proposed that directly learn regression functions to link appearance features to 3D structure [EL04], [SKM05], [AT06], [RFZ04] [BM09].

We have found that the peculiarities of our domain necessitate an approach that is different from prior work in computer vision. Talented animators often purposely violate the geometry of the human body, using subtle squash and stretch to convey emotion or muscle activity. To create secondary motion, it is necessary to track these changes in body shape and reconstruct them plausibly in 3D, rather than filter them out as noise.

3. Approach

In this section, we present our algorithm for creating plausible 3D secondary motion that is driven by the motion of a hand-animated character. We estimate the driving signal in three dimensions, build collision volumes that interact with the simulated 3D elements, and composite the rendered scene elements with the hand-animated drawing while maintaining relative depth ordering.

3.1. Three-dimensional Driving Signal

The frames that are drawn by the artist contain only the perspective view of the animated character. As a result, we are faced with two types of ambiguity—the depth ambiguity and the composite motion ambiguity. The depth ambiguity occurs because multiple 3D trajectories can yield the same 2D projected trajectory (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 secondary motion (e.g a simulated ponytail) is added, choosing the correct interpretation is essential or the ponytail will not have the correct dynamic motion.

We now describe how we resolve the composite motion ambiguity by registering a motion capture sequence, and

© The Eurographics Association 2010.

then back-project sparse markers on the 2D drawing to generate the driving signal in three dimensions.

3.1.1. User Input and Preprocessing

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 with four markers each. 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. These two user inputs are designed to fit into the traditional 2D animation workflow [Cul90, JT95]—the 'dots' can be marked when the cleanup or inbetweening artist re-touches every frame of the animated sequence, and the color segmentation can be done as part of the ink and paint process without requiring additional effort.

We also ask the user to select a motion capture segment that has similar depth information as the hand-drawn sequence when viewed from the same point of view. The 3D poses in this motion capture segment provide z-depth information, thus allowing us 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. This motion capture segment can differ from the hand-animation in timing because we preprocess the segment via the Dynamic Time Warp algorithm [SC90, Ell].

3.1.2. Registration

We register the poses of the time-warped motion capture segment (from now on, called 'motion capture poses') with the hand-drawn poses by estimating a projection matrix $\mathbf{M}$—thus, any movement in the markers over and above the movement of the motion capture poses is attributed to the camera. For each frame i, the user-specified virtual markers for the hand-drawn frames are $\tilde{\mathbf{x}}_i = [\tilde{x}_1, \tilde{y}_1, \tilde{x}_2, \tilde{y}_2 ... \tilde{x}_N, \tilde{y}_N]^T$. The 3D marker positions for the motion capture poses are $\tilde{\mathbf{X}}_i = [\tilde{X}_1, \tilde{Y}_1, \tilde{Z}_1, 1, \tilde{X}_2, \tilde{Y}_2, \tilde{Z}_2, 1, ... \tilde{Z}_N, 1]^T$, expressed in homogeneous world coordinates. We compute across a moving window of K frames around the frame i to increase robustness.

The primary objective for good registration is minimizing the geometric projection error,

$$e_p = \sum_{t=-K/2}^{K/2} ||\tilde{\mathbf{x}}_{i+t} - \mathbf{M}\tilde{\mathbf{X}}_{i+t}||_2.$$

Because this projection matrix is also going to be used to render the 3D simulated elements, we must include domain-specific constraints: skew and tilt are assumed to be zero, the scale factors are computed from the image resolution, the focal length is pre-specified. These assumptions are similar to Hornung and colleagues [HDK07] and Petrovic and colleagues [PFWF00]. The remaining unknown parameters are denoted $\rho(i) = (\theta_x(i), \theta_y(i), \theta_z(i), t_x(i), t_y(i), t_z(i))^T$.

Other domain-induced constraints are that the renderable camera should be above ground level, $e_g = (t_z - \mu)$, roll should be minimum, $e_r = |\theta_y|$, and the camera should move smoothly, $e_s = ||\rho(i) - \rho(i-1)||_2$.

Finally, we estimate $\rho^*(i)$ such that

$$\rho^*(i) = \underset{\rho}{\operatorname{argmin}}(\omega_1 e_p + \omega_2 e_g + \omega_3 e_r + \omega_4 e_s), \qquad (1)$$

where $\omega_1, \omega_2, \omega_3$ and ω_4 are the associated weights.

3.1.3. Back-projection

Once we have the perspective projection operator $\mathbf{M}$ obtained by registration, we look for 3D points that will project exactly onto the user-specified virtual markers $\tilde{\mathbf{x}}_{ij}$ under the action of $\mathbf{M}$. At any frame i, each marker j can be represented in homogeneous world coordinates as $\mathbf{X}^w_{ij} = [X^w_{ij}, Y^w_{ij}, Z^w_{ij}, 1]^T$. Then,

$$\tilde{\mathbf{x}}_{ij} \cong \mathbf{M}\mathbf{X}^w_{ij}.$$

We can rewrite this congruence relation, using the Direct Linear Transform (DLT) algorithm [HZ03], as

$$\tilde{\mathbf{x}}_{ij} \times \mathbf{M}\mathbf{X}^w_{ij} = 0. \qquad (2)$$

On rearranging the cross product as a matrix operation,

$$\mathbf{C}\mathbf{M} \begin{bmatrix} X^w_{ij} \\ Y^w_{ij} \\ Z^w_{ij} \\ 1 \end{bmatrix} = 0, \qquad (3)$$

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. m_i^T are the rows of the projection matrix $\mathbf{M}$.

Now, it is sufficient to estimate the z-depths in order to completely determine the corresponding 3D points. We assume the z-depth for each marker to be equal to the corresponding value in the motion capture poses $\tilde{X}$. For the i^{th} frame,

$$m_3^T \tilde{\mathbf{X}}_{ij} = m_3^T \mathbf{X}^w_{ij} \qquad \forall j = 1, ..., N. \qquad (4)$$

We add normalization constraints to fix the scale factors in homogenous coordinates to unity.

$$[0, 0, 0, 1]\, \mathbf{X}^w_{ij} = 1 \qquad \forall j = 1, ..., N. \qquad (5)$$

Stacking the constraints in Equations 3, 4 and 5 yields a linear system for each frame,

$$\mathbf{A}_{ij}\mathbf{X}^w_{ij} = \mathbf{b}_{ij}. \qquad (6)$$

Finally, we add a smoothing term by minimizing $||\mathbf{X}^w_{ij} - \mathbf{X}^w_{(i+1)j}||_2$, for all virtual markers j. We solve for the least

© The Eurographics Association 2010.

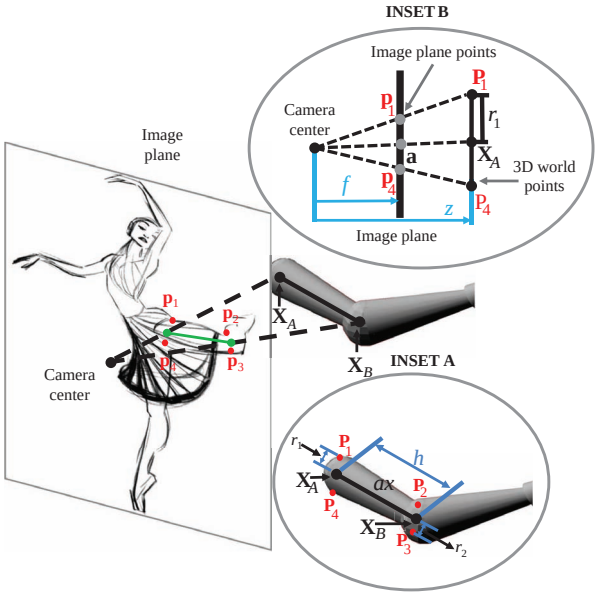

Figure 3: *User-specified markers are back-projected to obtain the 3D marker positions* $\mathbf{X}_A$ *and* $\mathbf{X}_B$. ***Inset A****: The cylindrical collision volume is characterized by its axis, height, and the radii of either face.* ***Inset B****: This inset describes how we compute the radius 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.*

squares solution to the following stacked linear system for a window of K frames,

$$\mathbf{W}\begin{bmatrix}\begin{bmatrix}\mathbf{A}_{11} & 0 & \dots & \dots \\ 0 & \mathbf{A}_{21} & \dots & \dots \\ . & . & \dots & \dots \\ . & . & \dots & \mathbf{A}_{KN}\end{bmatrix} \\ \begin{bmatrix}\mathbf{I} & -\mathbf{I} & 0 & \dots \\ .. & \mathbf{I} & -\mathbf{I} & \dots \\ 0 & \dots & \dots & \dots \\ 0 & \dots & \mathbf{I} & -\mathbf{I}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\mathbf{X}^w_{11} \\ \mathbf{X}^w_{21} \\ \dots \\ \mathbf{X}^w_{KN}\end{bmatrix} = \begin{bmatrix}\mathbf{b}_{11} \\ \mathbf{b}_{21} \\ \dots \\ \mathbf{b}_{KN} \\ \mathbf{0} \\ \dots \\ \mathbf{0}\end{bmatrix}, \tag{7}$$

$$\mathbf{W}\mathbf{A}_{full}\mathbf{X}^w_{full} = \mathbf{b}_{full}, \tag{8}$$

where $\mathbf{W}$ is the weight matrix that describes the relative weights between the geometric constraints and the smoothing terms.

3.2. Collision Volumes

In order to create believable interaction with complex simulations such as cloth, we fill in the space between the virtual markers with collision volumes (Figure 3), modeled as V tapered cylinders connected by spherical joints. Each cylinder must project to the image plane bounding box $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$ for that limb. These bounding boxes are obtained as part of the user input though we provide a simple algorithm to provide a good approximation—this routine looks for the boundary as defined by the first black pixel in the direction perpendicular to the line joining markers A and B. In cases where the arm crosses the torso for example, the routine incorrectly marks the boundary of the arm as the torso boundary. These cases are corrected by the user.

In this section, we will explain the details for one limb, and we will drop indices for clarity. Intuitively, we back-project the quadrilateral $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$ to the same z-depth as the 3D markers A and B (Figure 3). Then, we spin it about the axis AB and the surface of revolution gives us the collision volume for the limb AB. The axis vector $\vec{ax}$ and its magnitude (height) h are determined from the 3D world positions ($\mathbf{X}_A$ and $\mathbf{X}_B$) for the markers A and B. Figure 3 illustrates these quantities.

Here, we discuss how to obtain the radii r_1 and r_2 for the faces of the tapered cylinder. Let $\mathbf{P}_q$ be the 3D world position for the image point $\mathbf{p}_q$, where $q = 1,2,3,4$. We solve linear equations for each point on the bounding box. The back-projection is written out using the DLT algorithm (as in Equation 3),

$$\mathbf{C}\mathbf{M}\mathbf{P}_q = \mathbf{p}_q. \tag{9}$$

The z-depth of $\mathbf{P}_q$ is assumed equal to the z-depth of the corresponding limb marker,

$$\mathbf{M}(3,:)\mathbf{P}_q = \mathbf{M}(3,:)\mathbf{X}_A \quad \text{for } q = 1 \text{ and } 4, \tag{10}$$

$$\mathbf{M}(3,:)\mathbf{P}_q = \mathbf{M}(3,:)\mathbf{X}_B \quad \text{for } q = 2 \text{ and } 3. \tag{11}$$

The final constraint sets the scale factor to unity,

$$[0,0,0,1]\,\mathbf{P}_q = 1. \tag{12}$$

The radii for the faces of the tapered cylinder are then computed as,

$$r_1 = \frac{\sqrt{||\mathbf{P}_1 - \mathbf{P}_4||_2}}{2},\ r_2 = \frac{\sqrt{||\mathbf{P}_2 - \mathbf{P}_3||_2}}{2}. \tag{13}$$

3.3. Simulation and Depth Compositing

The frame-by-frame 3D markers and collision volumes can be imported into any 3D modeling and animation software package (Figure 4). As 3D scene elements, they can interact with other objects in the scene. For example, in Figure 4(b), the 3D jumping jacks character is imported into a Maya scene. An artist has created pompoms and a skirt. The Maya dynamics engine is used to physically simulate the motion of the pompoms and the skirt, and their interaction with the 3D collision volumes of the hand-drawn character [Sta09]. Maya is also used for rendering. In addition to the 'beauty' pass which contains the scene elements with texture, lighting etc, we also render the depth map and the occlusion map (Figure 4(c-e)).

© The Eurographics Association 2010.

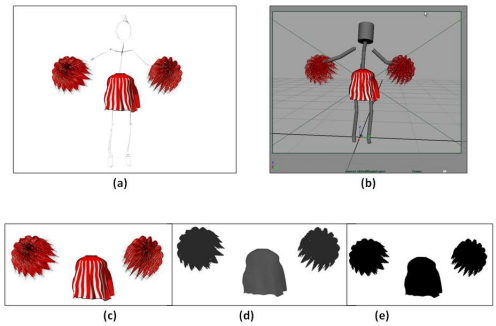

Figure 4: *(a)The final composited frame with the hand-drawn character and the rendered 3D elements. (b)Collision volumes imported into the 3D animation package. (c)rendered scene elements. (d)z-depth for the rendered elements. (e)occlusion map for the rendered elements.*

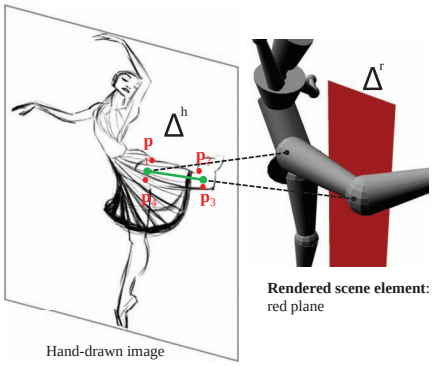

Figure 5: *Our method generates an alpha map for the hand-drawn image that maintains depth ordering between the hand-drawn pixels and the rendered 3D scene elements.*

3.3.1. Depth Compositing

In order to fully integrate a hand-drawn frame Υ_i^h with the rendered scene elements Υ_i^r, they must be composited while maintaining depth ordering. The depth map Δ_i^r for the rendered scene element is obtained from the renderer (Figure 4). The depth map Δ_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 3.1.1). For stick figures, we segment out the dark pixels by thresholding inside an oriented window along the limb v.

The z-depth values for the pixels $\tilde{\mathbf{x}}$ (that is, the pixels corresponding to the N virtual markers) are known. Therefore, for each limb v ($v = 1,2,..V$), the depth values for its two end-points are known (Figure 5). Let $\mathbf{l}$ denote the line joining the end-point markers for limb v, whose image positions are $\tilde{\mathbf{x}}_a = (a_x, a_y)$ and $\tilde{\mathbf{x}}_b = (b_x, b_y)$. Then, $l = \frac{\tilde{\mathbf{x}}_b - \tilde{\mathbf{x}}_a}{||\tilde{\mathbf{x}}_b - \tilde{\mathbf{x}}_a||_2}$. Every pixel $\tilde{\mathbf{p}} = (\tilde{p}_x, \tilde{p}_y)$ 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 to obtain the depth Δ_i^h, and then scale it to match the units of Δ_i^r.

Table 1: *Summary of results*

Motion	Skinned character	Moving camera	3D simulation
Ballet	✓	✓	cloth
Twirl	✓	×	cloth
Jumping jacks	×	×	cloth, hair
Walk across	×	×	cloth, rigid bodies

Let η_i^h be the occlusion map for the hand-drawn frame, and η_i^r be the occlusion map for the rendered scene elements. The alpha matte α for the hand-drawn frame Υ_i^h is defined as the inverse of the gray-scale value. Because scanned drawings are in RGB format, they are first converted to grayscale [GW02]:

$$\Upsilon^h_{gray} = 0.298\Upsilon^h_{i\,R} + 0.587\Upsilon^h_{i\,G} + 0.114\Upsilon^h_{i\,G}, \quad (14)$$

$$\alpha = (255 - \Upsilon^h_{gray})/255. \quad (15)$$

To incorporate depth ordering, for all pixels p where $\eta_i^h(p) = 1$,

$$\alpha = \begin{array}{ll} \alpha & \text{if } \Delta_i^h < \Delta_i^r, \\ 0 & \text{otherwise.} \end{array} \quad (16)$$

The final composited image I_{final} is

$$I_{final\,i} = \alpha\Upsilon_i^h + (1-\alpha)\Upsilon_i^r. \quad (17)$$

4. Results

We have applied our method to four hand animated characters—a ballet dancer, a goofy character doing jumping jacks, a character doing a stylized walk across the screen, and a little girl twirling.

In the ballet example, shown in Figure 6, two scarves were attached to the wrists of the dancer. The cloth simulation is driven by the three-dimensional trajectory of the wrist markers, and interacts with collision volumes for the body of the ballerina.

Our method can be used to create delicate effects like the strands of the pompoms in Figure 8, which would be fairly time-consuming to hand animate with a comparable degree of detail. We can also simulate effects such as cloth, particles and rigid bodies (the skirt in Figure 1, the snow and colored balls in Figure 7, and the puddle in Figure 8). The motion is physically plausible, and tweaking the secondary motion is just a matter of tuning parameters as with any simulation.

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

© The Eurographics Association 2010.

Figure 6: *Scarves are simulated as 3D cloth. An intricate pattern can be texture-mapped at no extra cost.*

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)
- Specifying 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 identical to the tuning required to add a visual effect to a 3D animation. Maya can be replaced by any other simulation engine.

We have also evaluated the robustness of our approach on a synthetic example to explore how close the motion capture sequence needs to be to the hand animated sequence. A motion capture walk sequence (normal walk in a straight line) is projected to 2D. The 2D markers are back-projected and z-depth is provided by five other 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 9 illustrates that broad jump and run have similar numerical error in depth, which follows because the depth ordering for the limbs is the same for both these actions when viewed sideways. These results show that a stylistically different walk (happy walk) captured on a different subject can also be used in our method, while a sequence with large difference in z-depth (curve walk) cannot.

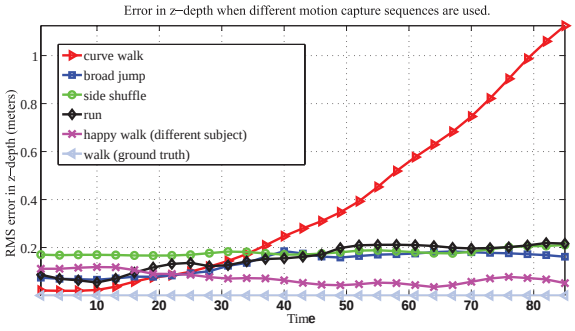

Figure 9: *We examine how different motion capture segments affect the error in z-depth. The normal walk is the ground truth. The z-depth error for the curve walk increases as the mocap poses veer towards the camera. The least error is seen in a happy walk sequence captured on a different actor and this motion could be used as a driving signal. Run and side shuffle have smaller errors than curved walk, but for detailed interaction, these motions would probably also not provide sufficiently accurate z-depth values.*

5. Discussion

We have presented a method to augment hand-drawn animation with the secondary motion of three-dimensional scene

© The Eurographics Association 2010.

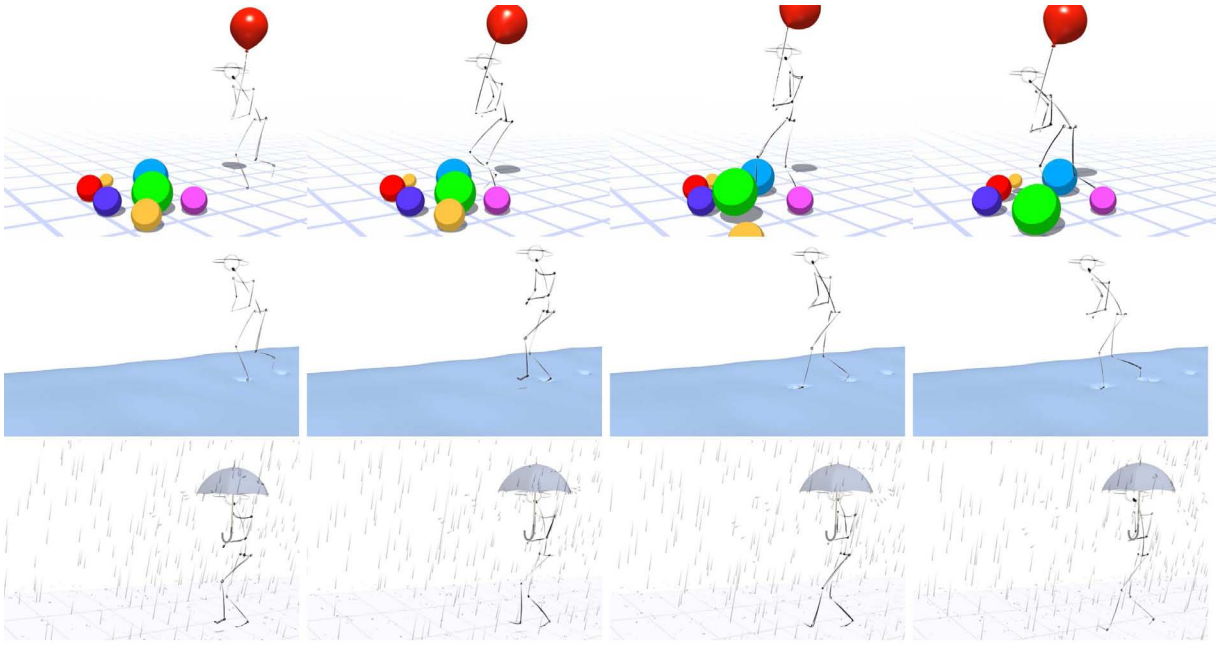

Figure 7: *Stylized walk across the screen. The dynamics of the balloon, its string, and the colored balls are driven by the motion of the character (top row); snow deforms as the character steps through it (middle row); and rain bounces off an umbrella attached to the wrist (bottom row).*

elements. The secondary motion is generated as a dynamic simulation. We first estimate the driving signal for the simulation in three dimensions, then build 3D collision volumes that align with the hand-drawings in image space, and finally, composite the rendered simulation with the original hand-drawn frames. We achieve this 3D augmentation by employing user input that can be accommodated in the existing animation workflow, and by using motion capture data of similar behaviors, a resource that is easily available today.

Because we need a contiguous mocap sequence to infer z-depth, that limits us to hand animations of human-like characters, and the amount of leeway provided by state-of-the-art time warping, retargeting, motion blending, and motion resequencing techniques [Gle98, LCR*02, BVGP09, IAF09, ZS09]. As illustrated in Figure 9, the motion need not be an exact match but must be similar. In our implementation, we use tapered cylinders as the collision volumes. Spheres and other simple shapes would be equally easy to register with the hand animation. An animation that required a complex shape (a tiger's snout for example) might be more difficult, especially if significant squash and stretch occurred in the hand animation. The techniques of Li and colleagues [LGXS03] could be used to create tighter collision volumes by altering 3D mesh geometry to conform to artist-drawn lines. The marker trajectories are computed without using non-penetration constraints on the attached collision volumes. As a result, it is possible for the collision volumes to interpenetrate. Simulating tight-fitting clothes on the hand-drawn character, such as a shirt with sleeves, would likely require penetration constraints to be built into the optimization in Equation 8.

The 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. 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.

While the 3D secondary motion created by our method is driven by the hand animation, the hand-drawn lines are not affected by the 3D elements of the scene. In other words, we have implemented a one-way coupling between the hand-drawn layer and the three-dimensional CG system [OZH00]. It would be interesting to explore techniques for two-way coupling. For example, ruffling the hand-drawn hair and skirt on the ballerina in response to the simulated scarves would add to the plausibility, or believability, of the augmentation.

Though all our results have used hand-animations as input, we could also augment video sequences—for example, a scarf on a person walking outdoors on a windy day. Just

© The Eurographics Association 2010.

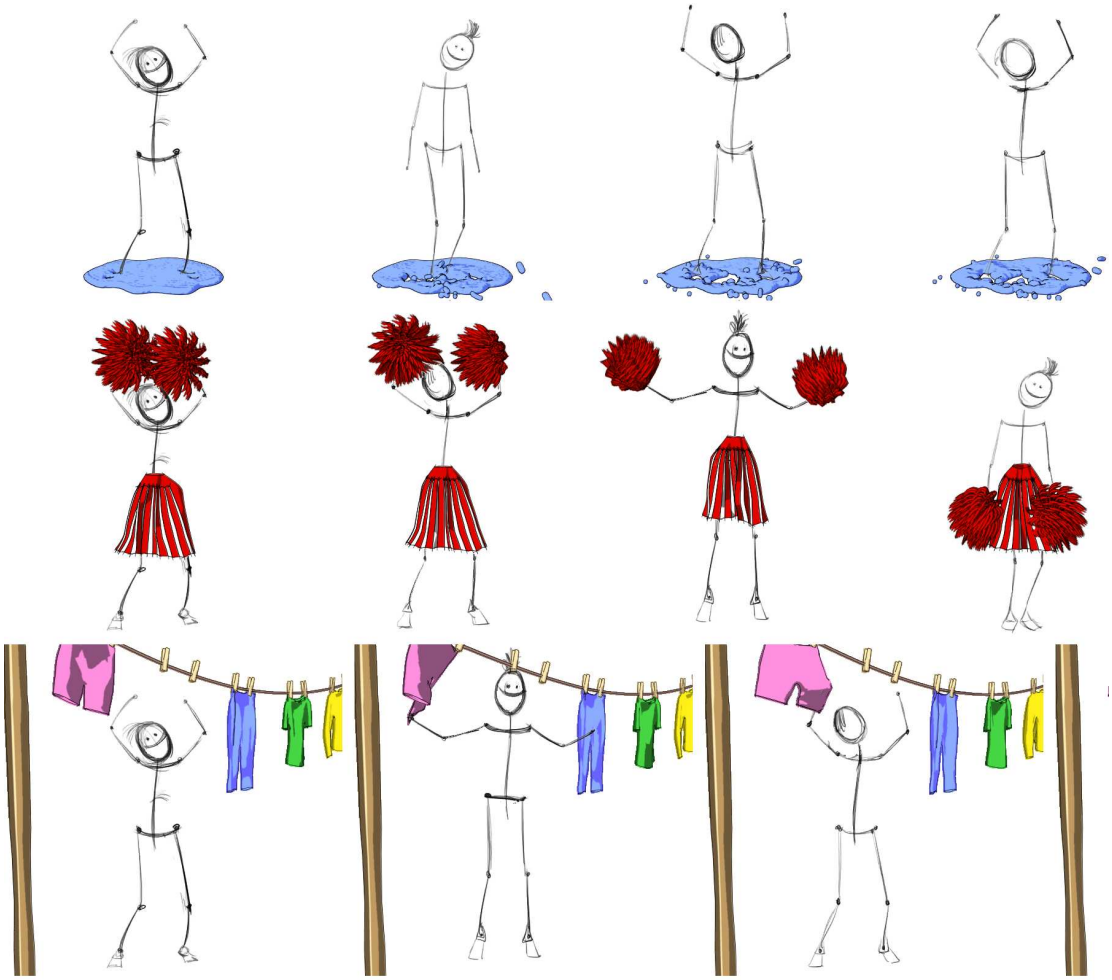

Figure 8: *Goofy character doing jumping jacks. Water splashes in response to the feet (top row); the pom-poms deform in a physically realistic way, and the skirt billows around the character's legs (middle row); and the character's hand interacts with clothes on the clothesline (bottom row).*

as we have matched the rendering style of the added 3D elements by using a toon shader, we could draw on the literature in the vision community on transferring lighting, shadows and other visual cues, so as to augment video data.

References

[Anj01] ANJYO K.: Bridging the gap between 2D and 3D: A stream of digital animation techniques. In *9th Pacific Conference on Computer Graphics and Applications* (2001), pp. 332–335.

[AT06] AGARWAL A., TRIGGS B.: Recovering 3d human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 1 (2006), 44–58.

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics 21*, 3 (2002), 399–407.

[BM98] BREGLER C., MALIK J.: Tracking people with twists and exponential maps. *IEEE Conference on Computer Vision and Pattern Recognition* (1998).

[BM09] BOURDEV L., MALIK J.: Poselets: Body part detectors trained using 3d human pose annotations. *IEEE International Conference on Computer Vision* (2009).

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM Transactions on Graphics 28*, 3 (2009), 36:1–36:6.

[CJTF98] CORRÊA W. T., JENSEN R. J., THAYER C. E., FINKELSTEIN A.: Texture mapping for cel animation. *ACM SIGGRAPH '98* (1998), 435–446.

[Coo02] COOPER D.: 2D/3D Hybrid character animation on "Spirit. *ACM SIGGRAPH '02 conference abstracts and applications* (2002), 133–133.

[Cul90] CULHANE S.: *Animation From Script to Screen*. St. Martin's Press, New York, 1990.

[DAC*03] DAVIS J., AGRAWALA M., CHUANG E., POPOVIC

© The Eurographics Association 2010.

Z., SALESIN D. H.: A sketching interface for articulated figure animation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), 320–328.

[Dan99] DANIELS E.: Deep canvas in Disney's Tarzan. *ACM SIGGRAPH '99* (1999), 200.

[DKD03] DEMIRDJIAN D., KO T., DARREL T.: Constraining human body tracking. *IEEE International Conference on Computer Vision* (2003).

[EL04] ELGAMMAL A., LEE C.: Inferring 3d body pose from silhouettes using activity manifold learning. *IEEE Conference on Computer Vision and Pattern Recognition* (2004).

[Ell] ELLIS D.: Dynamic time warp(DTW) in Matlab. http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/.

[FAI*05] FORSYTH D. A., ARIKAN O., IKEMOTO L., O'BRIEN J., RAMANAN D.: Computational studies of human motion: part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision 1*, 2-3 (2005), 77–254.

[Gle98] GLEICHER M.: Retargetting motion to new characters. *ACM SIGGRAPH '98* (1998), 33–42.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIC Z.: Implicit surface joint limits to constrain video-based motion capture. *ACM Transactions on Graphics 23*, 3 (2004), 522–531.

[GW02] GONZALES R., WOODS R.: *Digital Image Processing*, 2 ed. Prentice Hall, 2002.

[HDK07] HORNUNG A., DEKKERS E., KOBBELT L.: Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics 26*, 1 (2007), 1:1–1:9.

[HUF04] HERDA L., URTASUN R., FUA P.: Implicit surface joint limits to constrain video-based motion capture. *European Conference on Computer Vision* (2004), 405–418.

[HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry*, 2 ed. Cambridge University Press, 2003.

[IAF09] IKEMOTO L., ARIKAN O., FORSYTH D.: Generalizing motion edits with gaussian processes. *ACM Transactions on Graphics 28*, 1 (2009), 1:1–1:12.

[Joh02] JOHNSTON S. F.: Lumo: Illumination for cel animation. *NPAR '02: Symposium on Non-Photorealistic Animation and Rendering* (2002), 45–52.

[JSH09] JAIN E., SHEIKH Y., HODGINS J. K.: Leveraging the talent of hand animators to create three-dimensional animation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009).

[JT95] JOHNSTON O., THOMAS F.: *The Illusion of Life: Disney Animation*. Disney Editions; Rev Sub edition, 1995.

[LC85] LEE H. J., CHEN Z.: Determination of 3d human body postures from a single view. *Computer Vision, Graphics, and Image Processing 30* (1985), 148–168.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (2002), 491–500.

[LGXS03] LI Y., GLEICHER M., XU Y.-Q., SHUM H.-Y.: Stylizing motion with drawings. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), 309–319.

[MG06] MOESLUND T. B., GRANUM E.: A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding 81*, 3 (2006), 231–268.

[OZH00] O'BRIEN J. F., ZORDAN V. B., HODGINS J. K.: Combining active and passive simulations for secondary motion. *IEEE Computer Graphics & Applications 20*, 4 (2000), 86–96.

[PFWF00] PETROVIĆ L., FUJITO B., WILLIAMS L., FINKELSTEIN A.: Shadows for cel animation. *ACM SIGGRAPH '00* (2000), 511–516.

[RBCS07] ROSENHAHN B., BROX T., CREMERS D., SEIDEL H.-P.: Online smoothing for markerless motion capture. *Pattern recognition – Proc. DAGM 4713* (2007), 163–172.

[RBCS08] ROSENHAHN B., BROX T., CREMERS D., SEIDEL H.-P.: Staying well grounded in markerless motion capture. *Pattern recognition – Proc. DAGM 5096* (2008), 385–395.

[RBS07] ROSENHAHN B., BROX T., SEIDEL H.-P.: Scaled motion dynamics for markerless motion capture. *IEEE Conference on Computer Vision and Pattern Recognition* (2007).

[RFZ04] RAMANAN D., FORSYTH D., ZISSERMAN A.: Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 1 (2004), 65–81.

[Rob98] ROBERTSON B.: Mixed media. *Computer Graphics World* (Dec. 1998), 32–35.

[SBF00] SIDENBLADH H., BLACK M. J., FLEET D. J.: Stochastic tracking of 3d human figures using 2d image motion. *European Conference on Computer Vision* (2000), 702–718.

[SBS02] SIDENBLADH H., BLACK M., SIGAL L.: Implicit probabilistic models of human motion for synthesis and tracking. *European Conference on Computer Vision* (2002).

[SC90] SAKOE H., CHIBA S.: Dynamic programming algorithm optimization for spoken word recognition. *Readings in speech recognition* (1990), 159–165.

[SKM05] SMINCHISESCU C., KANAUJIA A., METAXAS D.: Discriminative density propagation for 3d human motion estimation. *IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[SSJ*10] SÝKORA D., SEDLÁČEK D., JINCHAO S., DINGLIANA J., S.COLLINS: Adding depth to cartoons using sparse depth (in)equalities. *Computer Graphics Forum 29*, 2 (2010), 615–623.

[ST03] SMINCHISESCU C., TRIGGS B.: Estimating articulated human motion with covariance scaled sampling. *IEEE Conference on Computer Vision and Pattern Recognition* (2003).

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. *IEEE International Conference on Computer-Aided Design and Computer Graphics* (2009), 1—11.

[Tar99] Tarzan. Walt Disney Feature Animation Studios, 1999.

[Tay00] TAYLOR C. J.: Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding 80* (2000), 349–363.

[UFF06] URTASUN R., FLEET D. J., FUA P.: Temporal motion models for monocular and multiview 3d human body tracking. *Computer Vision and Image Understanding 104*, 2 (2006), 157–177.

[WC09] WEI X., CHAI J.: Modeling 3d human poses from uncalibrated monocular images. *IEEE International Conference on Computer Vision* (2009).

[WFH*97] WOOD D. N., FINKELSTEIN A., HUGHES J. F., THAYER C. E., SALESIN D. H.: Multiperspective panoramas for cel animation. *ACM SIGGRAPH '97* (1997), 243–250.

[WHY03] WU Y., HUA G., YU T.: Tracking articulated body by dynamic markov network. *IEEE International Conference on Computer Vision* (2003).

[ZS09] ZHAO L., SAFONOVA A.: Achieving good connectivity in motion graphs. *Graphical Models 71*, 4 (2009), 139–152.

© The Eurographics Association 2010.