Interval Research Corporation Technical Report 1998-045; An HTML version of this document is available at `http://www.interval.com/papers/1998-045`. Result image sequences are viewable at `http://www.interval.com/papers/1998-045/results/`

Hybrid rigid and non-rigid image-based modeling of articulated figures

Aaron Hertzmann*†
Trevor Darrell*

(*) Interval Research Corporation
(†) Media Research Lab, Dept. Computer Science, New York University
hertzman@mrl.nyu.edu, trevor@interval.com

Abstract

We describe a new method for rendering new poses of articulatd figures. The long-term goal of this work is to take images or video of a person in different poses, and then render images of that person in new poses. The human body is modeled as a rigid kinematic chain. A morphing technique models non-rigid changes in shape and texture of each kinematic link. This system achieves much more generality than either pure rigid or pure non-rigid models could achieve alone.

We demonstrate results on planar motion of a single arm. From only two images of the arm, the algorithm can generate a wide range of believable poses. We believe that the new approach is powerful enough to model 3D motion of a complex, articulated figure.

1 Introduction

This report addresses the problem of modeling the appearance of human bodies. Current computer graphics techniques for photorealistic human modeling are difficult and expensive, and are thus only feasible for high-end production, such as [14, 19]. Furthermore, it is nearly impossible to produce truly convincing images that do not appear computer-generated — imagine trying to generate convincing animation of actors based on old footage. For many applications, such as teleconferencing and video games, an efficient and effective modeling technique is desirable.

Our method is based on ideas from image-based rendering. The figure is conceptually modeled as a kinematic chain, used to parameterize the rigid figure motion. Each body segment is rendered by non-rigid morphing of the example images. These deformations are accurately represented, because the morphing is done in an aligned frame of reference, similar to the rectification phase of View Morphing [20].

The idea of this work is to combine the advantages of both rigid and non-rigid transformations. The body is modelled as a rigid, articulated figure. However, the deformation of each segment is modeled with a Radial Basis Function (RBF). We rigidly transform each example for a segment to the same orientation, and then do RBF interpolation. (RBF interpolation can be thought of simply as morphing, with particular choices for the interpolation coefficients.) This means that the RBFs only model non-rigid shape/texture changes, and do not attempt to model rigid changes in orientation or position. The appeal of this method is to use each technique for what it is good at: rigid transformations model the overall rigid transformations of the segments, and RBFs pick up the slack by modeling the additional non-rigid transformations. This method also has the potential to properly decouple unrelated body segments for interpolation (i.e. moving the arm doesn't cause the leg to move).

All of the image analysis in this work were done by hand. However, many of the subproblems (segmentation, articulated tracking, dense correspondence) are areas of continuing work in computer vision (e.g. [7, 3, 4]). Many of the steps of the analysis could be done automatically or by an assisted system.

2 Related Work

Most image-based rendering algorithms render new views of real-world objects from example images[1] [13]. The most prominent of these algorithms generate new views of static 3D environments, e.g. [15]. These static geometry techniques do not model changes in shape or texture.

Morphing [2, 12], linear interpolation, and radial basis functions (RBFs) each model non-rigid deformations, by producing linear combinations of shape and texture from example images. These methods alone are inappropriate to model poses of articulated figures, since they do not capture important constraints such as constant-length limbs. Darrell [8] has demonstrated a clustering algorithm for selecting appropriate example images for pure non-rigid interpolation. Although our work requires more modeling and image analysis, it requires far fewer source images and can potentially extrapolate over a much wider parameter space. Our work could also be used as the rendering module together a clustering method.

Rigid and non-rigid methods have been combined to model human faces [1, 18, 5]. In these systems, the overall rotation and translation of the face is modelled rigidly, and the local deformations of shape is modeled non-rigidly. Our work is similar. We model each segment of the body with a separate rigid transformation. View Morphing [20] also separates a single rigid motion from changes in shape. This philosophy of our work is also similar to Debevec et al. [9], where an assisted rigid analysis is combined with view-dependent texturing to generate new views of architecture.

Teichmann [23] creates articulated skeletons from a single 3D dataset.

3 Preliminaries

All points and vectors are expressed in homogeneous coordinates. In 2D, points and vectors are represented as $[x, y, 1]^T$, and $[x, y, 0]^T$, respectively. 3D points and vectors may be represented with an extra depth (z) component, or with a disparity component; 3D transformations must be computed accordingly.

[1] Other uses of IBR include accelerating the rendering of 3D scenes and object recognition.

For disparity maps, a rigid object motion can be simulated with a rigid camera motion (see Appendix B). The presentation in the remainder of this paper applies equally to 2D (planar) or 3D transformations.

We represent the appearance of a body segment as a matrix called an *image list*: $A = \begin{bmatrix} S \\ X \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & \dots \\ t_1 & t_2 & t_3 & \dots \end{bmatrix}$, where each $t_i = [r_i, g_i, b_i]^T$ contains texture information. This representation is similar to a layered depth image [22], but is more convenient for expressing point correspondences. (Layered depth images allow efficient incremental computation during reprojection.)

4 Image Analysis

The input images must first be analyzed and annotated. There are several steps to the analysis.

1. **Correspondence**: Dense correspondence between the image points on the figure is computed between each frame.

2. **Segmentation**: The figure is separated into kinematic links. The segmentation must be compatible with the correspondence (i.e. the same segmentation is used for each example image).

3. **Linkages**: The corresponding axes for each segment is identified in each frame. Each axis consists of a pivot point and a direction. The direction is indicated by an end point, although the absolute position of the end point is only important if the end point is used as the pivot point for another link.

In our experiments, these steps were done with two input images which were annotated by hand-labelled contour correspondences and split into two segments.

Using the contour points, dense correspondence between contours is generated in a standard way. A Delaunay triangulation of the contours is generated for each segment, followed by a mapping from each pixel in one triangle to the corresponding pixel in the corresponding triangle in the other image. Note that some contours may have a different Delaunay triangulation than the corresponding contour; we pick the triangulation that has better aspect ratios in both images. Triangle correspondence is computed as follows: We begin with a triangle (p_0, p_1, p_2) in image 1, and the corresponding triangle in image 2 (q_0, q_1, q_2) expressed in homogeneous coordinates. To find the corresponce q in image 2 for a point p in image one, we compute the barycentric coordinates of p in triangle one, and the image coordinates of q in triangle 2: $q = [q_0, q_1, q_2][p_0, p_1, p_2]^+ p$. The pseudoinverse is used in case of degenerate triangles.

We then annotate the input image with articulated segments in the plane. A fist, elbow, and shoulder point were each selected from the image contours. These points are in correspondence, and the elbow point represents the pivot point of the forearm. All of the image points in the forearm half of the segmentation belong to the fist-elbow axis, and all of the remaining points belong to the elbow-shoulder axis.

5 Training

Let e_1, f_1, s_1 be the positions of the fist, elbow, and shoulder in image 1; e_2, f_2, and s_2 are the corresponding points in image 2. These are 2D homogenous image points in our experiments; they could also be 3D homogeneous points for images with depth or disparity. We have found that rendering is not sensitive to the precise locations of these points, and approximate tracking or annotations are sufficient — non-rigid interpolation accounts for this. This is encouraging.

Let $\vec{F} = e_1 - s_1$, $\vec{S} = f_1 - e_1$. All training angles for the arm will be placed in a matrix Θ, where each row corresponds to an example image. $\Theta_{i,1}$ is the angle between $\vec{S}$ and $\vec{F}$ (i.e the angle of the forearm relative to the angle the upper arm segment): $\Theta_{i,1} = cos^{-1} \frac{\vec{S}^T \vec{F}}{||\vec{S}||||\vec{F}||}$. $\Theta_{i,2}$ is the absolute angle of the upper arm segment: $\Theta_{2,i} = cos^{-1} \frac{[0,1,0]\vec{S}}{||\vec{S}||}$. Additional example images for the arm are added as additional rows.

The following treatment shows the math for the forearm; the upper arm is treated in the same way.

The example images and dense correspondence are converted to image lists to make correspondences explicit. Each image point from the forearm in image 1 is represented as a vector $p = [x, y, 1, r, g, b]^T$, expressing the image position and the texture value of the point. These points are combined into a matrix: $P = [p_1, p_2, p_3, ...]$. The corresponding points are placed in another matrix: $Q = [q_1, q_2, q_3, ...]$. The matrices are ordered so that p_i in image 1 corresponds to q_i in image 2.

For each segment, we first align the examples. Let T be a transformation that aligns the forearm in image 2 to the forearm in image 1: $Te_2 \doteq e_1$, $T(f_2 - e_2) \doteq (f_1 - e_1))$ In other words, the two elbows coincide and the fists point in the same direction.[2] Let be Q' be the Q aligned: $Q' = \begin{bmatrix} TS \\ X \end{bmatrix}$ where $Q = \begin{bmatrix} S \\ X \end{bmatrix}$ (S is the shape component of Q and X is the texture). Any additional example images will be aligned to the same axis.

P and Q are now used as training data for an RBF. Let $\vec{P}$ and $\vec{Q}$ be the vectorized forms of P and Q ($\vec{P} = [p_{11}, p_{21}, ...p_{12}, p_{22}...]$. The RBF training equation is

$$C\Phi = D$$

where $D = \begin{bmatrix} \vec{P} \\ \vec{Q} \end{bmatrix}$, $\Phi_{i,j} = \phi(||\Delta(\Theta_i - \Theta_j)||)$. This last term quantifies the distances between the parameters values between the two images: $\Theta_i = [\theta_{i,1}, \theta_{i,2}]$ as defined above. The difference between two scalar angle values must be computed with respect to cycles; we define $\Delta(a) = |\text{mod}(a, 2\pi) - \pi|, \Delta([a, b, c, ...]) = [\Delta(a), \Delta(b), \Delta(c), ...]$. $|| \cdot ||$ denotes the L2-norm (Euclidean distance). We use $\phi(x) = x$ (a linear RBF). Additional example images are rectified as above and added as extra rows to the training matrices (θ and D.)

---

[2] T is the composition of:

1. A translation by $-e_2$
2. A rotation by the angle between $(f_2 - e_2)$ and $(f_1 - e_1)$
3. A translation by e_1

Note that it is not necessarily the case that $Tf_2 \doteq f_1$.

We solve for the coefficients with

$$C = \Phi^{+} D$$

The two body segments use the same training parameters Θ. In general, each link in the chain should use training parameters corresponding to it's relative orientation, as well as the orientation of any connected links.

6 Synthesis

Synthesis is conceptually specified by new fist/elbow/shoulder positions. Our current method for computing new fist/elbow/shoulder positions from user input is described in Appendix A

Given the new fist/elbow/shoulder positions and their joint angles Θ, it is straightforward to synthesize new images. The shape and texture of the new images are interpolated with:

$$\vec{N} = C\vec{r}$$

where $\vec{r_j} = \phi(||\Delta(\Theta - \Theta_j)||)$. We then reshape $\vec{N}$ to N, using the same dimensions as P and Q. N represents the new shape and texture of the limb.

Finally, we transform N to the new elbow/fist or fist/shoulder position as before. The transformation consists of a rotation to align the rectified limb direction with the new direction, and a translation to align the rectified pivot point to the new pivot point. The same synthesis process is applied to any other segments, and all of the segments are composited in an image.

The final transformation could also include a global camera transformation. For 3D images, this would allow viewing the new figure from any angle.

In our current system, many holes appear in these appears due to sampling problems — these are the same holes that appear in any simple forward mapping implementation. Additionally, a small fissure sometimes appears at the elbow joint. A hole-filling algorithm is used to eliminate holes. Our current hack just creates a blurred version of the source image, and then fills all 0's in the source with the values from the blurred image. This method has the side-effect of creating some unnecessary halo artifacts around the segments.

7 Experiments

See `http://www.interval.com/papers/1998-045/results/` for an example of rendering articulated arm images using our hybrid rigid and non-rigid system. Source images are displayed, together with annotations depicting the arm contour, kinematic chain, and segmentation into parts. (These were generated by hand using a semi-automated tracking system.) Dense correspondences were automatically generated based on the tracked contours, and each example image was aligned to a global frame for RBF training. Two example arm images were used, one upright and one outstreched. Movie files show the interpolated result generating images for intermediate arm locations. As can be seen in comparision to methods which use only rigid or only non-rigid interpolation, our method correctly captures the appearance of the articulated chain from just two annotated example images.

8 Future Work

The extension of these ideas to 3D and to multiple segments is conceptually straightforward, although there may be many pitfalls. For example, in the two segment case, it is not necessary to use the absolute orientation as part of the training parameters for the forearm shape. However, the texture values don't match between the two segments if different training parameters are used (one parameter instead of two). The challenge will be to maintain the continuity of appearance over the body even when different sets of training angles are used for different segments.

The example shown in this paper is equivalent to warping both example images to the target orientation and then linearly interpolating shape and texture. However, the RBFs provide a convenient formalism for warping many example images together.

References

[1] S. Avidan, T. Evgeniou, A. Shashua and T. Poggio. Image-Based View Synthesis by Combining Trilinear Tensors and Learning Techniques. *ACM VRST'97* Conference.

[2] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH '92 Proceedings).* Vol. 26, July, 1992. pp 35–42.

[3] Christoph Bregler and Jitendra Malik. Tracking People with Twists and Exponential Maps. *Proc. CVPR-98*, Santa Barbara, CA, IEEE CS Press, 1998.

[4] Christoph Bregler. Learning and Recognizing Human Dynamics in Video Sequences *Proc. CVPR-97*, San Juan, Pureto Rico, IEEE CS Press, 1997.

[5] Christoph Bregler, Michele Covell, Malcolm Slaney. Video Rewrite: Driving Visual Speech with Audio. *SIGGRAPH 97 Conference Proceedings.* August, 1997. pp 353–360

[6] William J. Dally, Leonard McMillan, Gary Bishop, Henry Fuchs. The Delta Tree: An Object-Centered Approach to Image-Based Rendering, MIT AI Lab Technical Memo 1604, May 1996.

[7] Trevor Darrell. A Radial Cumulative Similarity Transform for Robust Image Correspondence, *Proc. CVPR-98*, Santa Barbara, CA, IEEE CS Press, 1998.

[8] Trevor Darrell. Example-Based Image Synthesis of Articulated Figures. To appear, *Neural Information Processing Systems 11*, MIT Press, 1998. available as `http://www.interval.com/papers/1998-020`.

[9] Paul E. Debevec, Camillo J. Taylor, Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. *SIGGRAPH 96 Conference Proceedings.* August 1996. pp. 11–20.

[10] Michael Gleicher and Peter Litwinowicz. Constraint-based Motion Adaptation. *The Journal of Visualization and Computer Animation.* 1998. Vol. 2, No. 8. pp. 65–94.

[11] Martin Jägersand, Image Based View Synthesis of Articulated Agents, *Proc. CVPR-97*, San Juan, Pureto Rico, pp. 1047-1053, IEEE CS Press, 1997.

[12] Michael Jones and Tomaso Poggio, Multidimensional Morphable Models, *Proc. ICCV-98*, Bombay, India, pp. 683-688, 1998.

[13] Sing Bing Kang. A Survey of Image-based Rendering Techniques. Technical Report 97/4. Digital Equipment Corporation Cambridge Research Laboratory, August 1997

[14] Chris Landreth. *Bingo*. Computer-animated short film. 1998.

[15] Leonard McMillan and Gary Bishop, Plenoptic Modeling: An image-based rendering system. *Proc. SIGGRAPH 95*, pp. 39-46, 1995.

[16] Leonard McMillan. An Image-Based Approach to Three-Dimensional Computer Graphics, Ph.D. Dissertation, University of North Carolina, April 1997.

[17] Tomaso Poggio and F. Girosi, A Theory of Networks for Approximation and Learning, MIT AI Lab Memo No. 1140. 1989.

[18] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski and David H. Salesin. SIGGRAPH 98 Conference Proceedings. July, 1998. pp. 75–84.

[19] Protozoa. ALIVE Performance Animation Software.

[20] Steve Seitz and Chuck Dyer, View Morphing. *Proc. SIGGRAPH 96*, pp. 21-30, 1996.

[21] A. Shashua. Trilinear Tensor: The Fundamental Construct of Multiple-view Geometry and its Applications. *International Workshop on Algebraic Frames For The Perception Action Cycle (AFPAC97)*, Kiel Germany Sep. 8–9, 1997.

[22] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, John Snyder. Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. *SIGGRAPH 96*, p75-82

[23] Marek Teichmann, Seth Teller. Assisted Articulation of Closed Polygonal Models. *SIGGRAPH 98 Technical Sketch.*

A Computing new fist/elbow/shoulder positions

Synthesis is specified by new fist/elbow/shoulder positions. In our interactive system, the shoulder is fixed, and the user clicks on a new fist position. The new elbow position is computed from the input. The relationship of the elbow to the fist and shoulder may be different for different joint angles, due to deformations. In the fixed elbow-shoulder-fist relationships we tried, the two body segments did not line up.

The problem is to find a consistent set of elbow/fist/shoulder positions that interpolate the test data. The linear RBF is not very good at extrapolating, so the elbow/fist/shoulder positions should ideally lie on the line between the examples in interpolation space. The line is parameterized by the joint angles, and passes through the endpoints. However, this space is nonlinear, and it is not immediately obvious how to directly compute the point on the line nearest to a given user input. An approximate method follows. We believe it is possible to create a more exact parameterization, but we have not yet implemented it.

Our current synthesis method is specified by new shoulder position s' and approximate fist/elbow positions (f',e'), which are used to derive approximate joint angles ($[\theta_1', \theta_2']$), as in Section 5.

A separate RBF is used to track the relative position of the elbow based on joint angles: $C_e\Phi = \begin{bmatrix} e_1 - s_1 \\ e_2 - s_2 \end{bmatrix}$. (This is the same Φ as in Section 5). This RBF only approximates the actual rigid and non-rigid computation of the elbow position. Fortunately, the elbow position undergoes little variation between the examples we have tested. For a given set of approximate angles, we compute the elbow position:

$$e'' = C_e\vec{r} + s'$$

where $\vec{r}_j = \phi(||\Delta(\Theta' - \Theta_j)||)$. The new fist is placed at $f'' = e'' + ||f_2 - e_2||\frac{f'-e''}{||f'-e''||}$. (Note that the distance between f'' and e'' is not important in the remaining synthesis, only the direction.) Finally, joint angles ($[\theta_1'', \theta_2'']$) are computed from the fist, elbow, and shoulder positions.

We have also tried computing all positions with an RBF: $C_{efs}\Phi = \begin{bmatrix} e_1 - s_1 & f_1 - s_1 & s_1 \\ e_2 - s_2 & f_2 - s_2 & s_2 \end{bmatrix}$. The motion is somewhat plausible, but less convincing in comparison to the other approach.

B Simulating Object Motion with Camera Motion

The image-based rendering problem of simulating camera movement for a fixed, Lambertian scene with depth/disparity/correspondence has been solved by several authors, such as [15, 22, 21, 13]. The motion of a single object with a fixed camera is exactly equivalent to the opposite camera motion around a fixed object. Therefore, we can use these IBR techniques as a black box for simulating 3D object motion. (Dally et al. [6] describe an efficient image-based object representation.)

Specifically, we assume that an IBR algorithm takes an image, a rotation (specified by a rotation matrix R') and a translation (specified by a vector $\vec{t'}$). The algorithm produces a new view of a scene, with the translation and rotation applied to the 3D camera position and orientation.

We want to use such an algorithm to simulate the appearance of an object rotating by R about some pivot point x, and translated by $\vec{t}$. The problem is: given R, x, and $\vec{t}$, find R' and $\vec{t'}$.

The desired translation is composed of four translations and rotations:

1. Translate the camera center from the origin to x.

2. Rotate the camera by R^{-1}.

3. Translate the camera center back to the origin.

4. Translate the camera center by $-\vec{t'}$.

(Recall that a object translation or rotation corresponds to the opposite camera transformation.) This gives a linear system:

$$
\begin{array}{rcl}
R'T(\vec{t'}) & = & T(-\vec{t})T(-\vec{x})R^{-1}T(\vec{x}) \\
\begin{bmatrix} R' & R'\vec{t'} \\ & 1 \end{bmatrix} & = & \begin{bmatrix} I & -\vec{t} \\ & 1 \end{bmatrix}\begin{bmatrix} I & -\vec{x} \\ & 1 \end{bmatrix}\begin{bmatrix} R^{-1} & \\ & 1 \end{bmatrix}\begin{bmatrix} I & \vec{x} \\ & 1 \end{bmatrix} \\
 & = & \begin{bmatrix} R^{-1} & R^{-1}\vec{x} - \vec{x} - \vec{t} \\ & 1 \end{bmatrix}
\end{array}
$$

where $T(\vec{t})$ is a translation by $\vec{t}$, and $\vec{x}$ is the vector from the origin to x. Solving, we get

$$
\begin{array}{rcl}
R' & = & R^{-1} \\
\vec{t'} & = & R^{-1}\vec{t} + R^{-1}\vec{x} - \vec{x}
\end{array}
$$

C Transferring Disparity Values

We have implemented image-based warping as described in [16]. (This implementation is not used in the examples shown in this report, since those examples are planar.) McMillan does not describe how to compute disparity values for the new image. Here is a derivation, following McMillan's notation in Chapter 3.

Let P_1 and P_2 be projection matrices for the original and target camera position, and t be the translation between the camera centers. ($P_2 = RP_1$, where R is the rotation between cameras.) Let r_1 and r_2 be the distances from a scene point $\dot{X}$ to each camera center. Let x_1 and x_2 be the image coordinates of $\dot{X}$. We therefore have:

$$\dot{X} = \dot{C}_1 + r_1 \frac{P_1 \bar{x}_1}{||P_1 \bar{x}_1||} = \dot{C}_2 + r_2 \frac{P_2 \bar{x}_2}{||P_2 \bar{x}_2||}$$

(This is similar to McMillan's Equation 3-2). Substituting the Equation before Equation 3-10 ($\delta(\bar{x}) = \frac{||P\bar{x}||}{r}$), and rearranging, gives:

$$\delta(\bar{x}_2)\left(\frac{P_1 \bar{x}_1}{\delta(\bar{x}_1)} - t\right) = P_2 \bar{x}_2$$

Taking the magnitude of both sides gives:

$$\delta(\bar{x}_2)||\frac{P_1 \bar{x}_1}{\delta(\bar{x}_1)} - t|| = ||P_2 \bar{x}_2||$$

$$\delta(\bar{x}_2) = \frac{||P_2 \bar{x}_2||}{||\frac{P_1 \bar{x}_1}{\delta(\bar{x}_1)} - t||}$$

Note that $||P_2 \bar{x}_2|| = ||RP_1 \bar{x}_2|| = ||P_1 \bar{x}_2||$, since R is orthonormal. (Rotating a vector doesn't change the magnitude.)

The LDI [22] formulation of image based rendering is much simpler (it is essentially based on depth), and can transfer depth much faster.