Tracking and Modifying Upper-body Human Motion Data with Dynamic Simulation

Victor B. Zordan and Jessica K. Hodgins

College of Computing and Graphics, Visualization, and Usability Center
Georgia Institute of Technology
Atlanta, GA 30332-0280
victor|jkh@cc.gatech.edu

Abstract. Character animations produced with motion capture data have many of the stylistic details seen in human motion while those generated with simulation are physically realistic for the dynamic parameters of the character. We combine these two approaches by tracking and modifying human motion capture data using dynamic simulation and constraints. The tracking system generates motion that is appropriate for the graphical character while maintaining characteristics of the original human motion. The system imposes contact and task constraints to add dynamic impacts for interactions with the environment and to modify motions at the behavior level. The system is able to edit motion data to account for changes in the character and the environment as well as create smooth transitions between motion capture sequences. We demonstrate the power of combining these two approaches by tracking data for a variety of upper-body motions and by animating models with differing kinematic and dynamic parameters.

1 Introduction

Subtle details in the motion of humanlike characters affect the believability, aesthetic, and impact of an animation or virtual environment. In this paper, we combine two approaches for generating motion, motion capture and dynamic simulation, with the goal of making it easier to animate a graphical character with physically realistic and natural-looking motion. Used separately, both of these approaches have advantages but also potentially serious flaws. Motion capture produces characters that move with the stylistic details of humans but the captured data is difficult to modify for new situations and characters. Dynamic simulation generates physically correct motion for characters that respond interactively in a changing environment such as a virtual environment or electronic game. However, the controllers required for simulated characters are difficult to construct because we cannot currently specify the details of human motion procedurally. To combine these two techniques, we use a dynamic simulation to track and modify motion capture data for human upper-body movements.

Our system uses motion capture data as the input to a tracking controller for a dynamic simulation. The simulation is created using the physical parameters of the animated character thus ensuring that the generated motion is physically correct for that character. Driven by the tracking controller, the simulation produces trajectories that resemble the input motion and maintain the style of the human data. For example,

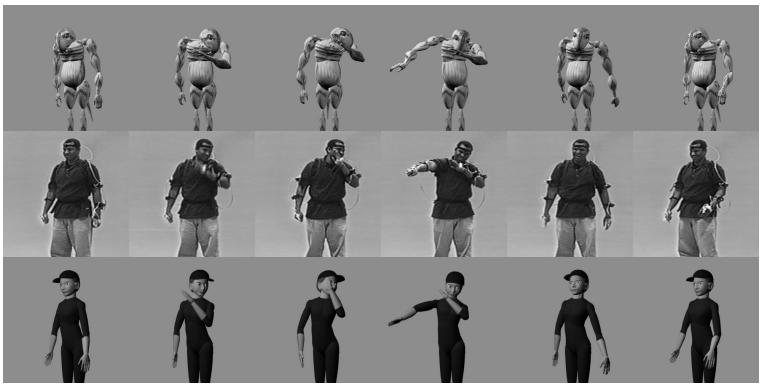


Fig. 1. Motion comparison – dynamic tracking vs. live motion. Tracking produces motion that maintains the overall characteristics but imposes the dynamics of the animated character.

Figure 1 shows a human actor and two animated characters tracking his motion. This system is used to perform a variety of gesturing and signaling behaviors for characters with several different body types.

Three additions to this basic system allow it to be used to adapt motion segments for new situations and to perform a richer variety of behaviors. First, a collision handler is added to generate realistic dynamic interactions with the environment. Second, specialized task controllers are added to edit character motion at the behavior level and to correct errors due to kinematic differences between the captured actor and the graphical character. Task controllers are also used to animate degrees of freedom that were not captured. For example, our system produces full-body motion from captured upper-body data by using a balance controller on the lower body. Finally, by modifying the input data, our technique adapts motions at a high level while relying on the simulation to maintain physical realism. For example, to ensure that contact occurs at each clap in a game of patty cake, the hand position is adjusted with inverse kinematics. Segments of motion can be re-ordered or scaled and smooth transitions can be created by modifying and interpolating between the segments and using the new sequence as input to the tracking controller. The resulting motion is physically plausible and tends to be smooth because it obeys a consistent set of physical constraints.

The next section of the paper reviews the relevant background literature, the third section describes the dynamic simulation and basic tracking system, the fourth section includes dynamic constraints and transitions with example implementations, and the last section concludes by evaluating the results.

2 Background

In this paper, we draw on research in two areas: generating motion using simulation and modifying motion capture data. Our simulations build on existing techniques for animating with dynamic simulations including hand-tuned control for rigid-body human-like characters [8, 1, 6] and automatic motion generation using dynamic models [19, 13, 15]. We rely on previous work in creating controllers, particularly the balancing techniques described by Wooten [21]. Bruderlin and Calvert animate human walking

by computing the basic motion with a simple simulation and adding extra degrees of freedom kinematically [3]. Their work and our work have similarities in that we both attempt to combine humanlike style with dynamics. However, they add biomechanical heuristics to enhance simulated motion while we use human data to control a fully simulated character.

Techniques that facilitate the use of motion capture data have received a great deal of attention in recent years because of the increasing availability and quality of the capture equipment. Most research in this area focuses on two key problems with motion capture: modifying or editing a captured segment to fit a particular character or situation and creating transitions between two separately captured segments.

Several researchers present techniques to adapt motion segments to new situations or characters. Witkin and Popović introduce a system that modifies time-warped joint trajectories according to keyframes using a scaling factor and an offset [20]. Gleicher and Litwinowicz use an interactive constraint solver for time-persistent constraints such as footholds [5]. Gleicher extended this technique to allow motion to be adapted to new characters and to two interacting characters [4]. Our approach has similarities with the recent work of Popović and Witkin who use a simplified dynamic model to edit motion for behaviors such as running and jumping [9]. However, unlike their work, we use a fully simulated character which allows fine control over interactions with the environment and emphasizes subtle differences caused by the character's dynamics. Other researchers take the approach of adapting motions by creating parametric behaviors from several sample sequences [14, 18, 10].

Transitions between motion capture sequences are required to make characters appear responsive in interactive virtual environments and electronic games. Witkin and Popović suggest creating transitions by interpolating joint angles [20]. Rose and his colleagues present an inverse dynamics formulation that generates transitions by minimizing the required torque [11]. Our work is similar in that we also use a dynamic model but we use forward dynamics and control algorithms rather than inverse dynamics and torque minimization.

3 Dynamic Simulation and Tracking Control

The goal of this research is to capture the subtle details that make motion appear natural or humanlike while maintaining physical realism. With this goal in mind, we choose to use upper-body motions as a testbed because people are particularly expressive with their arms and head. Free-space gestures are primarily stylistic while other upper-body motions are more purposeful and require that the appropriate contacts be made or positions and orientations maintained. Our basic technique uses a dynamic simulation of a humanoid and a tracking controller to follow human motion data. This system is appropriate for gestures or situations where there is a good kinematic match between the captured actor and the graphical character.

We use two types of dynamic simulations in the examples presented in this paper: an upper-body model and a full-body model. With the first model, only the upper body of a humanoid is simulated and legs are drawn graphically (Figure 2). The second model is a full-body simulation where the lower body is controlled with a balance controller. Depending on whether the upper-body simulation includes an articulated back, eight or nine rigid links are connected with three degree-of-freedom revolute joints for a total of 24 or 27 controlled degrees of freedom. The full-body simulations include dynamically

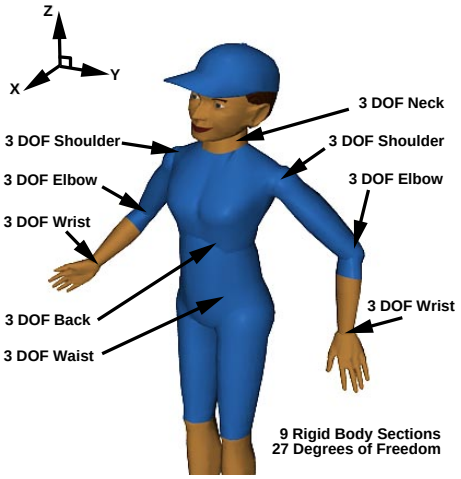


Fig. 2. Dynamic degrees of freedom. This model has 27 controlled degrees of freedom. with static, graphical legs. Another upper-body model has 24 controlled degrees of freedom excluding back articulation. Our full-body model has 48 controlled degrees of freedom with the additional joints in the legs used to maintain balance.

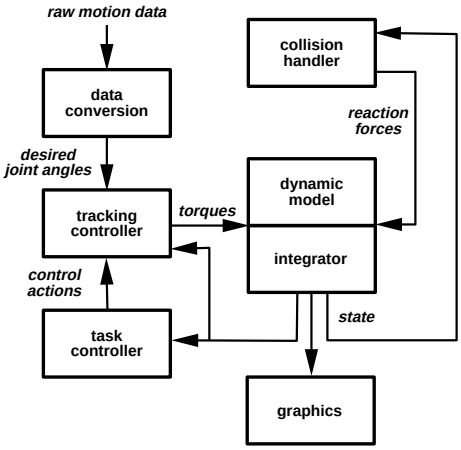


Fig. 3. Tracking system layout. Raw motion data is converted into continuous joint angles and used as the desired values for a tracking controller. The controller calculates torques for the dynamic model which is then integrated to generate motion. A task controller and collision handler may be added to achieve more complex and directed behaviors.

simulated legs and feet and have 16 rigid links with a total of 48 controlled degrees of freedom. Mass and moment-of-inertia information for the dynamic models is generated from the graphical body parts and estimated density values [6]. The equations of motion are calculated using a commercial solver, SD/Fast [12].

The upper body of these dynamic models is driven by a trajectory tracking control system as shown in Figure 3. Human motion data is converted to joint angles and used as the desired values for the tracking controller. The controller calculates torques based on the error between the state of the system and the desired joint angles. The resulting torques are applied to the dynamic model and the equations of motion are integrated forward in time to compute the new state of the system. Task-level control may also be included in the simulation as input to the control system. If necessary, collision forces are also applied to the dynamic model. Because of the expense of computing collisions, they are computed only for those body parts that are expected to collide in a particular behavior while other inter-body collisions are ignored.

Trajectories for the desired joint angles, $\theta_{desired}(t)$, are calculated from raw marker data and hierarchical skeletal information. The electro-magnetic Flock of Birds capture system provides global orientation data for each marker and body part at the rate of 15 to 25 Hz. The included orientation, expressed as a rotation matrix, is calculated for each joint:

$$\Theta_{desired} = \Theta_{ii}{}^{\top}\Theta_{io} \tag{1}$$

where Θ_{io} and Θ_{ii} are the orientation matrices of the outboard and inboard bodies at joint i for a particular sample in time. The outboard body is the next body moving

outwards in a branching tree rooted at the pelvis. The inboard body is the previous body encountered in the tree. The joint angle data is interpolated to create a set of continuous trajectories. We experimented with both spline and quaternion slerp interpolation and found that each worked satisfactorily with the finely sampled input motion data.

Like other systems that use joint controllers for dynamic behaviors [8, 6, 16], our system uses a proportional-derivative servo to calculate the low-level control torques at each joint:

$$\tau = k\ (\theta_{desired} - \theta_{actual}) - k_d\ (\dot{\theta}_{actual}) \tag{2}$$

where θ_{actual} and $\theta_{desired}$ correspond to the actual and desired joint angles, $\dot{\theta}_{actual}$ is the actual joint velocity, and k and k_d are constant gain and damping terms. This proportional-derivative servo, a simple spring and damper, acts as a first-order muscle model. We do not set explicit joint limits in the controller because the tracked human motion should not contain unexpected joint limit violations.

Gain and damping terms are determined according to an overall stiffness for the character. Individual gains are calculated from the overall stiffness by scaling according to the moment of inertia of the outboard body. Thus, the stiffness for the shoulder joint is scaled by the moment of inertia of the upper arm. Damping gains are one-tenth of the corresponding stiffness gain. We set the overall stiffness to a nominal value initially by hand and then modify it by a gradient search to minimize the error, ϵ:

$$\epsilon = \sum ||\theta_{desired} - \theta_{actual}||. \tag{3}$$

By minimizing this error, the tracking controller is tuned to produce motion that more closely matches the input data for a given simulation time step. This basic system is used to generate a variety of motions such as the gesturing and signaling behaviors in Figure 7 (see Appendix).

4 Dynamic Constraints for Task and Environmental Interaction

Adding dynamic constraints to the basic system allows the user to edit motion by controlling forces or motor actuation in a dynamic model. In particular, we implemented constraints that perform dynamic contacts and task-specific control. We consider two illustrative example behaviors: a hand-drumming motion as a dynamic contact constraint problem and a staff-pumping motion as a task constraint problem.

4.1 Environmental Constraints

Constraints that enforce environmental conditions have been implemented with inverse kinematics solvers in other systems, but some high-speed contacts are more appropriately handled with dynamics. For example, an inverse kinematics solution for drumming would place the hand on the drum at the moment of contact and restrict the hand from penetrating the drum. However, the timing, duration, and reaction of the impact would have to be crafted by hand. In contrast, a dynamic model enforces the hand position constraints by explicitly calculating the dynamic reaction forces corresponding to the drum impact (see Figure 8 in Appendix).

We implement collision detection and response using algorithms from O'Brien, Zordan, and Hodgins [7]. Collisions between the rigid geometry of the hand and the

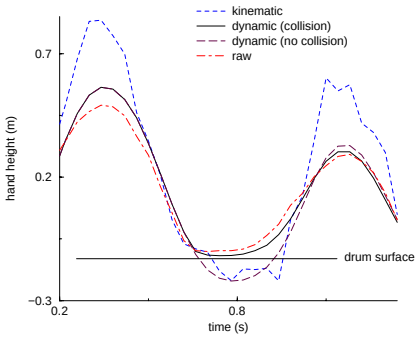


Fig. 4. Vertical position of a drumming hand. The dynamic data is shown with the drum/hand collision constraints on and off. The dynamic model without collisions and the kinematic model allow the hand to penetrate the drum noticeably.

drum are detected with a bounding-box hierarchy. Reactions are calculated based on penalties for position and velocity errors. The corresponding forces and moments are applied to the hand. The resulting motion will be modified by the collision forces to create a believable impact.

Figure 4 shows hand height data during a particular impact from several sources: our system with, and without, the collision constraints activated; the playback of the input joint angles through the kinematic model of the animated character; and hand-marker position data collected at the time of capture. Because our system uses only the orientation data, the recorded position data may be used for comparison if the kinematics of the actor and the character are similar.

4.2 Task Constraints

Although free-space gestures generally work well independent of kinematic differences between the human actor and the graphical character, other, more directed, tasks require a special purpose control system. As an example, we captured a human subject moving a staff up and down in a rhythmic fashion. When the raw motion data is played through a kinematic model, the staff does not move vertically and has unnatural accelerations. If the hand and staff are constrained using inverse kinematics, the staff may appear massless because the hand motion does not reflect the effort required to accelerate the staff. To solve this problem, we add a task controller for the wrist. Instead of using the human data as the desired value for the wrist joint, a proportional-derivative servo in the wrist uses feedback to maintain an upright orientation for the hand and staff. The rest of the body moves according to the original motion. The control gains for the wrist are easily hand-tuned so that disturbances are minimized and the staff sways in a realistic way under its own mass (Figure 8, see Appendix).

Task controllers can also be used to animate degrees of freedom that were not originally captured by using hand-designed control systems. In the bowing behavior in Figure 8 (see Appendix), motion capture data is used to drive the upper body of the character while the lower body maintains balance. The hips, knees, and ankles are controlled by a balance controller that holds the location of the center of mass over the feet [21].

5 Modifying the input trajectories

By modifying the input trajectories, our system can be used to edit motion data to account for mismatches between the character and human actor and adapt the motion to a variety of new situations. In this section, we describe examples in which input trajectories are modified using inverse kinematic offsets and interpolation for combining motion segments.

5.1 Inverse kinematic offsets

In the previous examples, we assume that the kinematic mismatch between the human actor and the graphical character is small or that the task is loosely specified. However, with larger kinematic errors, the tracking control system may fail to accomplish the task. In this case, we modify the human data with offsets computed using inverse kinematics. For example, the original motion data for the animation of children playing patty cake in Figure 9 (see Appendix) was recorded from an adult actor. When the motion is played through the child's kinematic model, the hands fail to meet where claps should occur. Specifying a location for the claps and using inverse kinematics to modify the trajectories fixes this problem.

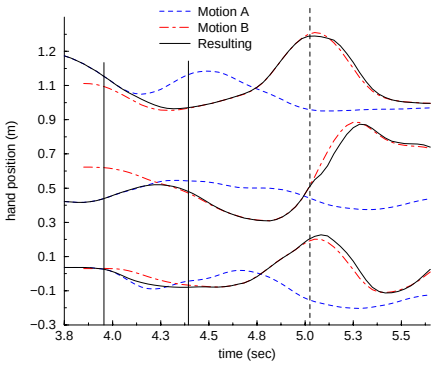
Our inverse kinematics solver re-positions the hand while maintaining its orientation by adjusting the shoulder, elbow, and wrist joints. The solver uses a Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm and a standard weighted evaluation scheme. A similar approach is discussed in more detail by Bodenheimer and colleagues [2]. At the intended moment of contact, the desired joint angles, θ_d, are equal to the inverse kinematics solution. The original joint angles are modified for a short duration before and after using an ease in/ease out interpolation. Quaternion slerp is used for interpolation [17].

Inverse kinematic offsets and a collision handler also allow us to modify the speed of the behavior while maintaining physical realism. The patty cake example can be scaled to 133 percent of real-time without changing the inverse kinematic offsets to create a behavior with physically realistic collisions at the new speed. For faster speeds, lags in the control system prevent contact from occurring. Lengthening the offset duration corrects this problem to some extent.

5.2 Combining motion segments

Another example of modifying the input motion is to combine two sequences into one by concatenating or interpolating and then use the new, longer sequence to drive the simulation. Creating realistic transitions is a common problem with motion capture libraries because data is captured in relatively short segments and combined in an on-line fashion to create an interactive character. One straightforward solution, often used in electronic games, requires choosing a particular *home* position and having each basis behavior start and end in this position. We apply this technique to the patty cake example by segmenting basis motions that begin and end with a character clapping her hands together. By concatenating re-ordered segments as input data, the system generates arbitrary clap sequences.

A more general system for creating transitions is to allow arbitrary segments to be connected independent of their starting and ending configurations. Our system does this by interpolating between the endpoints of the two segments and creating one, longer



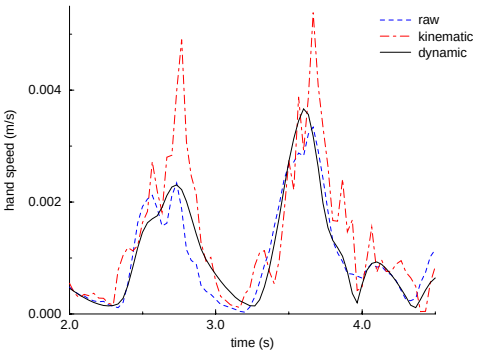


Fig. 5. Hand position during transition. The solid vertical lines delineate the bounds of the transition from the original sequence (*motion a*) to the slapping sequence (*motion b*). The dashed vertical line indicates when the hand hits the face and impact forces affect its trajectory.

Fig. 6. Hand motion comparison for gesturing. The dynamic model uses the same orientation data as the kinematic model but the dynamically tracked motion has speed profiles and peaks that are more similar to the raw marker data.

segment which is then used as the desired joint angles for the simulation. Similar to Witkin and Popović [20], we blend between two sequences of joint angles to create new joint trajectories. However, unlike their technique, these trajectories are used as desired values for the tracking controller. The new input data, $\theta_r(t)$, is created by shifting two sequences, *motion a* and *motion b*, so that they are lined up on a common timeline, *t*, and interpolating from $\theta_a(t)$ to $\theta_b(t)$ as:

$$\theta_r(t) = \theta_a(t)(1-\omega(t)) + \theta_b(t)\omega(t), 0 \le \omega(t) \le 1, t_0 \le t \le t_1. \quad (4)$$

The weighting parameter, $\omega(t)$, uses a sinusoid ease in/ease out function. Times t_0 and t_1 correspond to the beginning and end of the transition and are specified by the animator. Quaternion slerp is used for interpolation.

Because the entire motion sequence is tracked with the same dynamic model, a consistent set of physical laws is enforced and the transition appears smooth. This approach to creating transitions is similar to that of Rose and his colleagues who use inverse dynamics to create transitions between fixed motion sequences [11]. Unlike their work, our approach does not prevent the original sequences from being modified by the simulation. Thus, the tracking system may continue to alter the motion after t_1 to compensate for the dynamics of the transition.

Figure 5 shows a graph of hand positions for a dramatic transition. The original patty cake sequence is modified to include a slap in the face as one child becomes frustrated with her partner. The slerp interpolation generates continuous desired joint trajectories and the dynamic simulation smoothes and adds dynamic effects that mimic how the physical character might perform this transition. Like previous techniques, skill in creating transitions relies on the animator's choice of parameters. Poor choices for the beginning and ending of a transition can lead to inter-body penetration and unnatural postures as well as transitions that happen too slowly or too quickly.

6 Evaluation and Discussion

We present a system that uses a dynamic model to track and modify captured data for upper-body behaviors along with several examples that demonstrate the power of this approach. Critical evaluation of this technique is important but challenging because of the difficulty in quantifying metrics such as naturalness and style.

One possible metric for assessing whether the style of the motion has been maintained is a comparison of hand motion in the raw marker data, the kinematic playback of the data, and the dynamic tracked data. Figure 6 shows a comparison of hand speed for the gesturing motion of the alien and actor seen in Figure 1. Raw position data, set aside at the time of capture, is compared to simulated motion and to kinematic playback of the orientation data. The simulated data contains peak values similar to the raw hand marker. In contrast, the kinematic playback generates spikes that represent unnatural accelerations.

Across numerous examples, we have observed that the motion resulting from this system has particular characteristics caused by the dynamics. For example, the resulting motion is smoother than the incoming data because the motion trajectories are introduced as desired joint angles via a control system. However, the smoothing of the input trajectories also means that some tasks may not be accomplished without inverse kinematic offsets. Furthermore, because the system tracks the data without looking ahead in the trajectory, it does not anticipate changes and there is a time lag in the generated motion. Although this lag may be a problem when trying to animate tightly choreographed motions, in general, it is small enough to be ignored. More fundamentally, the control system can only track what was recorded, that is, the actual motion achieved by the human actor, not the actor's desired motion.

We present several modifications that incorporate behavior-specific information in situations where the basic system would not perform well. However, these examples represent only a preliminary step towards the general problem of extracting pertinent information from motion capture data to create new behaviors. In some cases, knowledge might be inferred from the motion capture sequence. For example, a small set of long jumps of various lengths should contain information about how the arm swing changes with the distance jumped. But, extracting information about more precise properties of a behavior such as balance is likely to be more difficult. In other situations, the required knowledge might not be present in the recorded data at all. For example, if we adapt a motion captured from an adult to a small child, the child's motion would have the precision of an adult rather than the increased variability expected in a child.

While the research presented here does not represent a final solution for combining motion capture data and simulation, we believe that it represents a promising step towards developing techniques that retain the important characteristics of dynamic simulation while extracting the stylistic details found in human motion data.

Acknowledgments

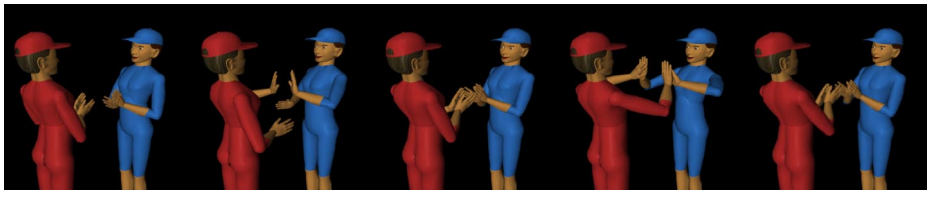
The authors would like to thank Chris Atkeson for his valuable feedback in this work, James O'Brien and Len Norton for use of their support software, Nancy Pollard and the OOB team for their part in *Alien Occurrence*, Scott Robertson for his efforts in recording the motion data, and Chad Jenkins, James O'Brien, Robert Orr, and Scott Myers for their help as motion capture talents.

References

1. Norman I. Badler, Cary B. Phillips, and Bonnie Lynn Webber. *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, New York, 1993.
2. Bobby Bodenheimer, Charles Rose, Seth Rosenthal, and John Pella. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, pages 3–18. Eurographics, Springer-Verlag, September 1997.
3. Armin Bruderlin and Thomas W. Calvert. Goal-directed, dynamic animation of human walking. In *Proceedings of SIGGRAPH '89*, volume 23, pages 233–242. ACM SIGGRAPH, July 1989.
4. Michael Gleicher. Retargeting motion to new characters. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 33–42. ACM SIGGRAPH, Addison Wesley, July 1998.
5. Michael Gleicher and Peter Litwinowicz. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation*, 9(2):65–94, 1998.
6. Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH '95*, pages 71–78. ACM SIGGRAPH, August 1995.
7. James O'Brien, Victor Zordan, and Jessica Hodgins. Combining active and passive simulations for secondary motion. Technical Report GIT-GVU-97-01, Georgia Institute of Technology, January 1997.
8. Dinesh Pai. Programming anthropoid walking: Control and simulation. Technical Report 90-1178, Cornell Computer Science, 1990.
9. Zoran Popović and Andrew Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 95*. ACM SIGGRAPH, August 1999.
10. Charles Rose, Michael Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.
11. Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH '96*, pages 147–154. ACM SIGGRAPH, August 1996.
12. Symbolic Dynamics Inc. *SD/Fast User's Manual*. 1990.
13. D. Tang, J. T. Ngo, and J. Marks. N-body spacetime constraints. *The Journal of Visualization and Computer Animation*, 6(3):143–154, 1995.
14. Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH '95*, pages 91–96. ACM SIGGRAPH, August 1995.
15. Michiel van de Panne and Eugene Fiume. Sensor-actuator networks. In *Proceedings of SIGGRAPH '93*, pages 335–342. ACM SIGGRAPH, August 1993.
16. Michiel van de Panne and Alexis Lamouret. Guided optimization for balanced locomotion. In *Computer Animation and Simulation '95*, pages 165–177. Eurographics, Springer-Verlag, September 1995.
17. Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1994.
18. Douglas J. Wiley and James K. Hahn. Interpolation synthesis for articulated figure motion. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pages 156–160, March 1997.
19. Andrew Witkin and Michael Kass. Spacetime constraints. In *Proceedings of SIGGRAPH '88*, pages 159–168. ACM SIGGRAPH, August 1988.
20. Andrew Witkin and Zoran Popović. Motion warping. In *Proceedings of SIGGRAPH 95*, pages 105–108. ACM SIGGRAPH, August 1995.
21. Wayne L. Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing Humans*. Ph.D. Thesis, Georgia Institute of Technology, 1998.

Simulated humanoid characters with different dynamics track human motion data to perform upper-body gesturing behaviors (Fig. 7).

Dynamic environmental and task constraints are used to modify motion capture data to generate impacts in a drum behavior (filmstrip spacing 0.13s), to keep astaff upright by controlling the wrist angles, and to add a lower body that balances in a bowing behavior (Fig. 8).



Simulations play patty cake by modifying the motion of one human actor using inverse kinematic offsets and tracking (filmstrip spacing 0.5s). Dynamic collisions occur between the hands at each clap (Fig. 9).