Hybrid Control For Interactive Character Animation

Ari Shapiro
University of California, Los Angeles
ashapiro@cs.ucla.edu

Fred Pighin
University of Southern California
Institute for Creative Technologies
pighin@ict.usc.edu

Petros Faloutsos
University of California, Los Angeles
pfal@cs.ucla.edu

Abstract

We implement a framework for animating interactive characters by combining kinematic animation with physical simulation. The combination of animation techniques allows the characters to exploit the advantages of each technique. For example, characters can perform natural-looking kinematic gaits and react dynamically to unexpected situations.

Kinematic techniques such as those based on motion capture data can create very natural-looking animation. However, motion capture based techniques are not suitable for modeling the complex interactions between dynamically interacting characters. Physical simulation, on the other hand, is well suited for such tasks. Our work develops kinematic and dynamic controllers and transition methods between the two control methods for interactive character animation. In addition, we utilize the motion graph technique to develop complex kinematic animation from shorter motion clips as a method of kinematic control.

1. Introduction

Kinematic control techniques such as those based on motion capture data can create very natural-looking animation. Complex human and animal activities can be captured and then animated on CG characters, making the characters move and behave realistically. In addition, these motions can be combined together using various blending and transition techniques to produce sequences of realistic motion. However, kinematic techniques are not suited for dynamically interacting characters or to produce dynamic effects. In order to do so, the kinematically controlled motion must be tweaked so as to maintain the appearance of dynamic interaction in addition to maintaining critical aspects of the original motion, often resulting in unnatural effects. Kinematic control techniques also must be designed for a specific environment and cannot easily to adapted to changing ones.

Physical simulation is a technique that is well suited to modeling interactive bodies since it automatically simulates dynamic effects between bodies and their environment. Dynamic control allows animated characters to both actively respond to a changing environment and produce physically-realistic effects by abiding to the laws of physics. Dynamic simulation and control can be employed to model complex physical interactions between animated characters and objects that would be impractical or time-consuming to produce kinematically. However, developing robust dynamic controllers that produce natural looking motion is in general a hard problem.

By combining kinematic control and dynamic control, the advantages of both techniques can be realized. Our system can switch between kinematic animation and physical simulation for any number of characters in the environment and determines which effects will be placed on each character. For example, it may be desirable to fully animate one character hitting another by motion captured data and only simulate the impact of the force of the hit on the second character. In other instances, all characters can be dynamically simulated. The characters can react to unexpected circumstances through the use of dynamic controllers which are automatically activated based on the state of the character and the environment. For example, a character falling down will automatically put its hands down to cushion the impact.

Our system uses both kinematic and dynamic control techniques at various stages for different objects in the animated scene where appropriate. The kinematic control and dynamic control for any object can be sequenced in any order and automatically composed together to produce effects not possible with either technique alone.

We build upon our previous work on dynamic controller composition [3]. Within our framework, controllers are now black boxes that can encapsulate any kind of animation technique. For example, they can be based on the direct application of motion capture, elaborate motion graph structures [7, 9] or dynamic controllers [5, 3].

The rest of the paper is organized as follows: related work is covered in Section 2. We describe the method of composing kinematic and dynamic controllers in Section 3. Controller activation is covered in Section 4. Section 5 details our implementation of the system while Section 6 outlines our results. Conclusions and future work are in Section 7.

2. Previous Work

Previous research for synthesizing complex animation has concentrated on either using kinematics and preserving the appearance of the dynamic aspects of the motion, or by physically modeling the interacting characters using dynamics. In addition, some hybrid approaches that combine the two methods have also been proposed.

Kinematic Techniques. Many techniques utilize the explicit and implicit information of key frames to produce plausible-looking character animation. Motions graphs [7] utilize a database of kinematic motion to capture transitions between movements and produce transitions between keyframes. The motion texturing technique proposed in [9] uses a stochastic model to capture the dynamic nature of kinematically controlled motion.

Trajectory and constraint based techniques such as [20, 11, 13, 15, 10, 16] employ physics as part of a constraint optimization problem. They are excellent tools for authoring motion, however, they cannot be used for interactive characters in their present form.

Dynamic Techniques. Developing dynamic controllers for complex articulated characters is not an easy task. [5] implement a series of elaborate dynamic controllers for human athletics. [8] develop a technique that can adapt balance and walking controllers to varying environmental conditions. There is also an interesting body of work that develops dynamic controllers for animals and other non-human characters [19, 18, 4]. One of the problems with dynamic control is that, often, it creates robotic motions that do not exhibit the fluidity of real human motion. [12] proposes an interesting technique to incorporate tension and relaxation aspects in the dynamic control of articulated characters. Finally, there is a large amount of work on dynamic controllers for humanoid robots in the robotics community that we do not review here for space considerations.

Hybrid Techniques. Certain techniques attempt to modify the kinematic motion by modeling dynamic effects and applying those changes to the original kinematic motion. In [22], a kinematically controlled boxer's head is adjusted from the force of a punch from another boxer. Also, a dynamic balance controller is used to maintain an upright position of a ping pong player. In [21], inverse kinematics are used to adjust the positioning of character arms in order to properly contact other objects and to place dynamic effects during contact with other objects, such as the effect on an arm of a character as it bangs on a drum.

Kinematics and dynamics have been combined in the past, however the effect was mainly to incorporate physical simulation along with kinematic control, rather than to have fully dynamically controlled characters. For example, characters impacted by a large force acted inanimately after being impacted by forces, thus achieving a physically plausible motion, but not responding as a "human-like" character would, such as grasping ones stomach after being hit in the midsection or preventing injury by cushioning ones body while falling to the ground.

Although, researchers have suggested before that combining dynamic and kinematic control would be useful, to our knowledge, our work is the first to combine dynamic and kinematic control in a uniform framework. Within our framework dynamic and kinematic control techniques can be incorporated as black boxes enhancing the motor control abilities of the character.

2.1. Composing Kinematic and Dynamic Controllers

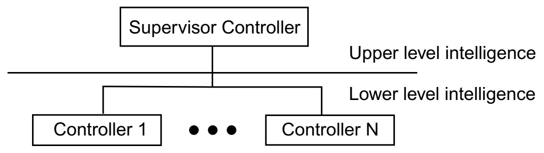

Figure 1. Two level composition scheme.

We build upon the composition framework presented in [3]. In our framework, *individual controllers* are black boxes encapsulating specialized control knowledge. First, an individual controller must be able to determine whether or not it can take the dynamic character from its current state to some desired goal state. Second, once an individual controller is active, it should be able to determine whether it is operating nominally, whether it has succeeded, or whether it has failed. Any controller that can answer these queries may be added to a pool of controllers managed by a *supervisor controller* whose goal is to resolve more complex control tasks. Hence, we have a two-level, hierarchical controller composition scheme, as illustrated in Fig. 1.

2.2. Dynamic Controllers

Controllers are activated and control the animated character as it passes through various states in the environment. Dynamic controllers are activated by examining the state of the character and the environment and bidding for control of the character in this state. In addition, dynamic controllers can utilize Support Vector Machines (SVMs) to determine success or failure from various states. The SVMs can be computed offline so that the dynamic controller can respond in real time to the state. Details are given in [3].

2.3. Kinematic Controllers

Kinematic controllers assume control of the character if they are able to the current state of the character with one of their keyframes. The matching is based on the *similarity value* which we compute by linearly combining the differences between the keyframe v and the state of the animated character:

$$s_i = \sum_{n=0}^{size} w_{ni} * \Delta q_{ni} + w_{mi} * \Delta v_{mi}, \tag{1}$$

where s_i is the similarity value at frame i, size is the number of DOFs, w_{ni} is the weight of the nth DOF of it ith frame, q_{ni} is the position of the nth DOF in frame i, w_{mi} is the weight for the velocity of the nth DOF in frame i and v_{mi} is the velocity of the mth DOF in the i frame. We compute the velocities using finite differences and spline interpolation.

If the *similarity value* is under a threshold, the kinematic controller will bid for control of the character. Global translation is not included in the similarity search, allowing the animated character to be positioned anywhere in space. When determining the *similarity value*, certain DOFs are weighted less than others. For example, the position of the DOF of the character's wrists is given less weight than the position of the DOF of the character's trunk. Since the hip and trunk affect larger parts of the body and have a greater influence on the state of the center of mass of the character, their state is more influential on the overall motion and they must given more weight than smaller part of the character.

Although a general similarity formula can be generated for all kinematic motions, specific weightings could be applied for each individual kinematic motion. For example, a kinematic motion of a character holding onto a horizontally-placed pole or climbing across monkey-bars would require exact positioning of the shoulders and hands, but allow flexibility in the positioning of the thigh, legs and feet since those DOFs are not restricted by contact constraints. We are investigating the use of an adaptive weighting scheme that tries to account for the particular nature of different classes of motion and we are looking for ways to incorporate contact constraints as part of the similarity metric.

2.4. Controller Activation

Our controllers can respond to both kinematic and dynamic control. The controllers are activated as a response to interaction between the character and its environment, automatically based on their pre-conditions, and following user instruction. Each animated character is under either kinematic or dynamic control at any instant in time. The system will incorporate hybrid techniques and constraint optimization methods in the future.

Control transitions can occur from any type of controller to any other. The following sections detail the four types transitions that can occur during animation and simulation:

1. Dynamic to Kinematic Transition.
2. Kinematic to Dynamic Transition.
3. Kinematic to Kinematic Transition.
4. Dynamic to Dynamic Transition.

2.5 Dynamic to Kinematic Transition

If a character, which is under physical simulation and controlled by a dynamic controller, switches to kinematic control, then we blend the current state of the character to the the new state dictated by the kinematic controller. Blending occurs over a short period of time (.5 seconds) and can be done either through linear interpolation of the difference between keyframes or via splines. Since the activation of a kinematic controller is based upon the *similarity value*(Section 2.3), the transition tends to look better for small *similarity values*. Increasing the *similarity value* threshold allows more flexibility but creates noticeable blending effects.

2.6. Kinematic to Dynamic Transition

Characters transition from kinematic to dynamic control upon contact with *active* objects. *Active* objects are those objects and characters that apply physical forces to each other that are not part of the original motion. Such forces may require a reaction that is different from the current motion and therefore the system has to re-evaluate its control strategy and possibly activate another controller. For example, two characters colliding with each other should respond to the forces that are subsequently imparted from the impact. A ball that is kicked by a character will apply a force on the character's foot and vice versa. In many cases, an object needs to be specified as *passive*, so that it will not triggered a response from the character. For example, ground contact is an integral part of a walking gait. When a kinematic walking controller is active the ground must be specified as a *passive* object so as to avoid triggering a response from the character. In contrast, a ball hitting the walking character on the head should be active in order to trigger a response.

In general, the implicit constraints of a kinematic motion must be extracted in order to determine which objects will be considered to be *active* or *passive* [14]. As another example, a kinematic controller that animates a character swinging a baseball bat must set the baseball bat object to be *passive* with respect to the character that swings it. Also, the *active* or *passive* state of the objects are relative to each object and are directed, where an object could apply its forces to another object but be unaffected by the force of the second object on it. The baseball bat in the above example is considered to be *passive* with respect to the character swinging it during kinematic control, and *active* for other characters and objects at all times. Objects are all considered to be *active* when the characters are under full physical simulation and only set to *passive* when they are part of the implicit constraints of a kinematically controlled character. Currently, there is no automatic method of extracting these constraints from the kinematic motion and this must be done manually.

The transition from kinematic to dynamic control will only occur if the kinematic controller fails to find a *similarity value* that meets the threshold during a similarity search. Thus, a high similarity threshold will prevent a character from transitioning to physical simulation and dynamic control yet still allow the animated character to apply forces to other objects. In this way, the effect of the character is similar to secondary motions [14] where an object imparts a force on another but it is not affected by the action-reaction effect.

Characters can also transition from kinematic to dynamic control as a reflex or protective response. For example, a dynamic controller that is activated when objects are moving towards the character's head will interrupt the kinematic controller to prevent injury under those conditions. Such reflex and protective controllers require constant evaluation of the state of the character and the environment.

Our system properly matches the positions and the velocities for the degrees of freedom of the characters when it switches between kinematic animation and dynamic simulation. Thus, falling or swinging animated characters will preserve their linear and angular momentums when switching to physical simulation.

2.6.1 Dynamic Tracking Controllers

In order to provide a natural-looking transition from kinematic control, each kinematic controller has a corresponding dynamic tracking controller. Dynamic tracking controllers make the character follow the kinematic motion, essentially acting as a follow through mechanism. In addition, the dynamic tracking controller provides a mechanism by which the original kinematically controlled motion can be adjusted by small interaction forces. Since it is difficult to robustly apply a force to a kinematically controlled posture and determine which aspects of the pose to retain[21] while simultaneously maintaining realism, we have simulate the entire motion and rely upon the robustness of the similarity matching and dynamic controllers to provide a plausible result.

2.7 Kinematic to Kinematic Transition

Transitioning between kinematic controllers is based on blending much like the transitions from dynamic to kinematic control. We blend the motions over a window of 15 frames based on a 30 frames per second sampling rate.

2.8 Dynamic to Dynamic Transition

Transitions between dynamic controllers can happen when the post-conditions of the first controller fall within the pre-conditions of the second controller [3]. Dynamic controllers maintain control of the character until they fall, succeed, the user specifies a new task or there is a need for a reflex or protective behavior.

2.9 Our Model

Animated human characters are articulated 3D skeletons with 38 degrees-of-freedom (DOF) that are modeled as a hierarchy of joints. Six (6) of the DOFs correspond to the global translation and rotation parameters. The models are equipped with natural limits on both the motion of the joints and the strength of the muscles [3].

2.10 Dynamic Simulation

Dynamic simulation is produced by SD/FAST, which produces optimized simulation code [6]. Collision detection is handled by the use of of SWIFT++ [2], a collision detection software package. Collision resolution is done via penalty methods that corrects geometry interpenetration using spring-and-damper forces. Impulse forces are applied to non-stationary objects under collision.

2.11 Kinematic Engine

Recent research in kinematic control [7, 1, 9] has shown that organizing recorded motions in a directed graph structure is a powerful paradigm for motion reuse. This process starts by segmenting a database of motions into a set of small fragments. The segmentation is performed by searching for good transition points between motions. The graph is built such that each path represents a motion with the similar quality as the recorded motions. The "motion graph" can then be used to synthesize novel motions: given a set of geometric constraint a search in the graph returns the best path candidate.

In our experiments, we implemented a system similar to the work by [7]. In this implementation the transition between the motions are the local minima of pair-wise motion disparity maps. To synthesize a new motion, we use as query a sequence of control points that defines a cardinal spline. The search is performed following a branch and bound strategy with limited horizon. At each step of the search the current motion candidate is compared to the input cardinal spline.

2.12 Software Environment

DANCE is a portable, extensible object-oriented modeling and animation system [17]. It provides a platform for implementing animation and control techniques with minimal design and implementation overhead.

The core system supports four base classes: Systems, Simulators, Actuators and Geometries. All System objects have physical and dynamic properties. Animated characters are modeled as ArticulatedObjects, a subclass of System that supports skeleton hierarchies.

DANCE utilizes a plug-in model, where objects of any type can be added to the system on-the-fly. Controllers are implemented as plugin subclasses of the Actuator core class.

3 Results

Our characters can react autonomously to changes in their environment and to user interaction as well as follow user instructions. The power of combining kinematic and dynamic control is demonstrated by the image sequences in this section.

A kinematically animated character performs a martial arts kick and sends his partner flying to the ground, Figure 3. The character that is hit reacts to the interaction using its available dynamic controllers.

A kinematically walking character trips over a ball and performs a dynamic roll over stunt in Figure 2. The roll over controller is parameterized such that it can perform variations of the roll over motion.

Figure 4 demonstrated a kinematically animated soccer player kicking a ball towards a dynamically animated goalie. The contact between objects and the motion of the ball is dynamically simulated. In order to generate the soccer player kicking the ball, the ball was placed carefully at

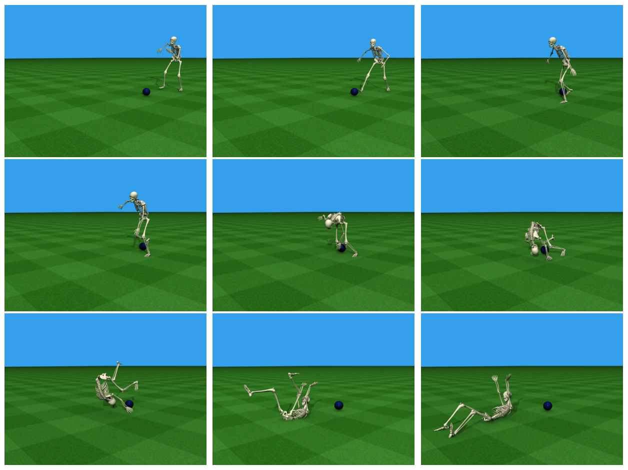
Figure 2. The animated character trips on a ball as it walks. Upon contact with the ball, the character switches from kinematic control to the use of a dynamic tracking controller. Once the character realizes that it is falling to the ground, it attempts to perform an athletic flip via a dynamic controller. Images are in raster order.

the proper point of contact with the foot. However, a more intelligent kinematic controller could use inverse kinematics to match the position of the foot with that of the ball, as shown in [21]. This improvement would require less interaction on the part of the animator.

4 Limitations

The kinematic controllers depend greatly on the quality of the original motion data and they suffer from all the problems associated with data-driven animation (scaling, sliding etc). The dynamic tracking controllers must occasionally be hand tuned to adjust the forces that are applied to particular DOF of the character in order to avoid jumps. We are investigating automatic ways to eliminate such problems.

The dynamic controllers for falling are part of previous work and are fairly robust. However, we would like to design a large library of controllers that can produce a wide range of reflexes, protective behaviors and voluntary motions.

5 Conclusions & Future Work

We have implemented a simple yet powerful framework that is capable of combining dynamic simulation and kinematic animation techniques in a unified fashion. Our system is capable of integrating current and future kinematic and

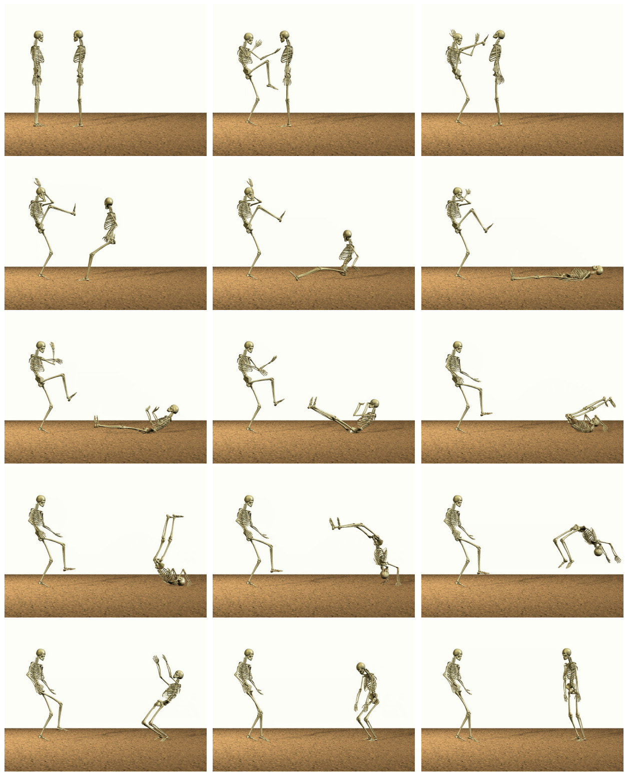

Figure 3. Kinematically controlled kick and dynamically controlled reaction and interaction. The motion of the character on the right is dynamically simulated. Images are in raster order.

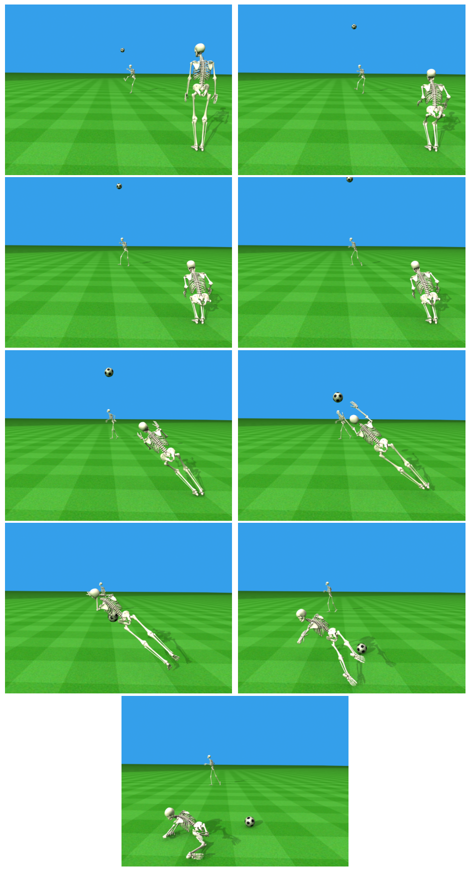

Figure 4. Kinematically animated soccer player, dynamic ball and goalie.

dynamic techniques since it imposes no restrictions on the structure of individual controllers. We have incorporated the motion graph technique [7] as an individual controller into our system. In addition, we plan to expand our system to include a variety of recently published animation techniques as separate controllers.

Our method does require, however, robust dynamic controllers in order to create plausible sequences of animations. Dynamic controllers currently exist to fall to the ground gracefully, jump, tumble and perform many ballistic motions. However, character balance using dynamic controllers is difficult and brittle under the influence of strong external forces. In addition, the creation of hand-tuned dynamic controllers can be laborious. Although some research has been done on learning dynamic control from kinematic motion, no established technique exists to do so robustly. We are currently investigating this problem.

We believe that our system closes a gap between kinematic and dynamic techniques. We have demonstrated that by leveraging the advantages of both techniques in a unified framework, we can produce complex motion and interactions for interactive characters.

6 Acknowledgements

We would like to thank Intel Corp. and Microsoft Corp. for their generous support through equipment and software grants.

This paper was developed with funds of the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

References

[1] O. Arikan and D. A. Forsyth. Synthesizing constrained motions from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[2] S. A. Ehmann and M. C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum*, 20(3):500–510, 2001.

[3] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In E. Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 251–260. ACM Press / ACM SIGGRAPH, 2001.

[4] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-based locomotion through control abstraction. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 63–70, August 1995.

[5] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 71–78, Aug. 1995.

[6] M. G. Hollars, D. E. Rosenthal, and M. A. Sherman. Sd/fast. Symbolic Dynamics, Inc., 1991.

[7] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[8] J. F. Laszlo, M. van de Panne, and E. Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996.

[9] Y. Li, T. Wang, and H.-Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, July 2002.

[10] C. K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416, July 2002.

[11] Z. Liu, S. Gortler, and M. Cohen. Hierarchical spacetime control. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 293–302, August 1994.

[12] M. Neff and E. Fiume. Modeling tension and relaxation fro computer animation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 81–88, July 2002.

[13] J. Ngo and J. Marks. Spacetime constraints revisited. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 343–350, August 1993.

[14] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins. Combining active and passive simulations for secondary motion. *IEEE Computer Graphics & Applications*, 20(4), 2000.

[15] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. P. Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 209–218, July 2000.

[16] Z. Popovic and A. Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM Press/Addison-Wesley Publishing Co., 1999.

[17] Removed for blind review.

[18] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. *Proceedings of SIGGRAPH 94*, pages 43–50, 1994.

[19] M. van de Panne and E. Fiume. Sensor-actuator networks. *Proceedings of ACM SIGGRAPH: Computer Graphics*, pages 335–342, August 1993.

[20] A. Witkin and M. Kass. Spacetime constraints. *Proceedings of ACM SIGGRAPH: Computer Graphics*, 22(4):159–168, August 1988.

[21] V. B. Zordan and J. K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation '99*, Sept. 1999.

[22] V. B. Zordan and J. K. Hodgins. Motion capture-driven simulations that hit and react. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 89–96, July 2002.