University of Pennsylvania
ScholarlyCommons

Center for Human Modeling and Simulation | Department of Computer & Information Science

11-1-1998

Motion Abstraction and Mapping with Spatial Constraints

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Ramamani Bindiganavale
University of Pennsylvania

Postprint version. Published in *Lecture Notes in Computer Science*, Volume 1537, Modelling and Motion Capture Techniques for Virtual Environments: International Workshop, CAPTECH'98, pages 70-82.
Publisher URL: http://portal.acm.org/citation.cfm?id=723374

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/hms/43
For more information, please contact repository@pobox.upenn.edu.

Motion Abstraction and Mapping with Spatial Constraints

Abstract
A new technique is introduced to abstract and edit motion capture data with spatial constraints. Spatial proximities of end-effectors with tagged objects during zero-crossings in acceleration are used to isolate significant events and abstract constraints from an agent's action. The abstracted data is edited and applied to another agent of a different anthropometric size and similar action is executed while maintaining the constraints. This technique is specifically useful for actions involving interactions of a human agent with itself and other objects.

Comments
Postprint version. Published in *Lecture Notes in Computer Science*, Volume 1537, Modelling and Motion Capture Techniques for Virtual Environments: International Workshop, CAPTECH'98, pages 70-82.
Publisher URL: http://portal.acm.org/citation.cfm?id=723374

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/hms/43

Motion Abstraction and Mapping with Spatial Constraints

Rama Bindiganavale and Norman I. Badler

Computer and Information Science Department
University of Pennsylvania, PA 19104-6389, USA

Abstract. A new technique is introduced to abstract and edit motion capture data with spatial constraints. Spatial proximities of end-effectors with tagged objects during zero-crossings in acceleration space are used to isolate significant events and abstract constraints from an agent's action. The abstracted data is edited and applied to another agent of a different anthropometric size and a similar action is executed while maintaining the constraints. This technique is specifically useful for actions involving interactions of a human agent with itself and other objects.

1 Introduction

When one person mimics the actions of another, the two actions may be similar but not exact. The dissimilarities are mainly due to the differences in sizes between the two people, as well as individual performance or stylistic variantions. In this paper, we address the first of these issues, namely, controlling virtual humans of various sizes who imitate the action of a motion captured subject.

A subject, called the *primary agent*, is motion captured executing a specific action. We are mainly interested in movements involving interactions with other objects (*e.g.*, drinking from a cup, digging with a shovel, etc.). The imitators, referred to as *secondary agents*, try to execute the same action by interacting with the objects in a similar fashion. As the body sizes and segment lengths of the primary and secondary agents are likely to differ, the motion imitation may not be successful or satisfactory. The thesis of this paper is that *relationships between the world, the body, and the end-effectors (hands, eyes) of the primary agent have been overlooked and are of considerable importance in reconstructing correctly scaled motions.* Often the objects being held are simply wielded for effect, such as holding a shield or slashing with a sword. Keeping feet in contact with the ground plane is one frequently encountered problem, but usually only vertical displacements are moderated: the actual horizontal step position may be input to inverse kinematics procedures to keep the body from floating or sinking. The issue of changing the step locations is related to the motion mimicry problem, but we do not address it here. In this paper, we assume that maintaining spatial constraints for hands and eyes – such as grasping a cup at the correct place and bringing it to the mouth for a drink – is more important than maintaining a similar trajectory or motion style between the various spatial constraints. Style integration will be considered in subsequent efforts.

Several types of motion editing techniques have been developed in the past few years for different purposes. Some of the techniques [1, 7, 21, 23] use signal processing methods to edit and modify the actions for the same agent. A few techniques [6, 15, 18, 19, 22] try to replicate the motions of a person on different sized avatars in real time. The techniques described in [9, 11] use optimization methods to modify the original motions in the presence of space-time constraints. As explained in [9], space-time refers to "the set of all DOF (joint angles and figure position) over the entire animation sequence." In [17], a new set of transition motions are created between two basis motions using space-time constraints. All these techniques treat the problems of mapping motions to other agents, of modifying the nature or style of the motions, and of modifying the motions while maintaining space-time or spatial constraints as separate problems and solve them individually. In [13], constraint based motions are adapted to other agents, but they do not consider interactions between objects and self. In [12], optimization techniques are used to retarget the motions to other agents during object interaction. But a very simple human model is used and the problem of visual constraints is not considered. In this paper, we introduce a technique to automatically recognize such spatial and visual alignment constraints from captured motions. Maintaining these constraints is the basis of motion mapping from the primary agent to secondary agents.

The problem of recognizing motion events directly from (synthetic) image sequences was first studied by Badler [4]. We update these notions to abstract information about significant events and spatial constraints from 3D motion captured data. We begin by generating motion for the primary agent from the motion capture data by using real-time optimization techniques [20, 24] to solve for the kinematic constraints imposed by the data itself. During the execution of the action, we use the concepts of *zero-crossing* and *co-occuring spatial proximities of end-effectors with interacting objects* to recognize the spatial constraints. We then modify the original motion to fit another agent of a different size while maintaining the same spatial constraints. We also abstract the line of attention of the primary agent during significant events. We then impose this as an additional spatial constraint to be solved during motion generation for the secondary agents. This alignment constraint forces the secondary agent to *look at* the same objects. This provides a very natural affect as, in general, people tend to look at an object while interacting with it [8].

The rest of the paper is organized as follows. Section 2 describes the human body model and the technique used to derive the motions for the primary agent from the motion capture data. Section 3 describes the technique to recognize the spatial constraints and Section 4 explains the method to compute the locations of the constraints. Section 5 describes the technique to map the motions to other agents and Section 6 describes a simple technique to recognize the visual attention of the primary agent. Section 7 presents the results and Section 8 discusses the impact of this approach.

2 Deriving Motions from Motion Capture Data

We have implemented this technique completely within the Transom *Jack*®[3] software. The human model we use is highly articulated and has 68 joints and 135 degrees of freedom.

The technique introduced in this paper can be applied to data from any source - motion capture, keyframe or procedural. We have chosen to use only motion captured data. We use the CyberGlove from Virtual Technologies and the MotionStar system from Ascension Technology. The CyberGlove has 24 sensors and generates the joint angles for all the fingers. The MotionStar system consists of one Extended Range Controller (ERC), one Extended Range Transmitter, and 12 Bird units, each controlling a single receiver (referred to as a sensor in the remainder of this paper). The two systems are calibrated separately but synchronized together to generate data at a frequency of 60Hz.

As a preliminary, off-line step in deriving motions from motion capture data for the primary agent, an avatar is built to the size of the subject and is calibrated by placing one of the sensors of the MotionStar system on the lower back of the subject roughly corresponding to the *L5* segment of the spine (sacro-iliac). In the human model, a corresponding site[1] (FOBpelvic) is created in the *L5* segment of the spine. The transformations between the MotionStar reference frame and the Jack reference frame are calculated by positioning the pelvic sensor at the FOBpelvic site. All the sensors are then mapped correctly onto the human model in the Jack environment. Next, using the data from all the sensors for the first frame, the human model is postured correctly to match the initial posture of the subject. Finally, sites are automatically created within the human model at the locations where the sensors lie on the body.

To generate the motions, kinematic constraints are established between the newly created sites corresponding to the sensor positions on the model and the sensors themselves. As the sensors move, the human model moves with them along trajectories computed subject to the constraints [5]. This process easily creates motions in real-time while interacting with an object for a similarly-sized avatar. To recreate the same motions for a different-sized agent while maintaining the spatial constraints, we first need to post-process the data to recognize the spatial constraints and map the newly derived data to the secondary agents.

3 Recognition of Spatial Constraints

During interactions with objects, the spatial constraints between the agent and the objects are in general defined by the proximities of the end-effectors and the objects. End-effectors correspond to sites in the human model that are constrained to follow the sensors in the motion capture system. Hence, assuming that the inverse kinematic routines solve the constraints exactly, the sensors themselves can be used to keep track of end-effector locations.

[1] Sites are oriented co-ordinate triples.

One method for recognizing spatial constraints is to use fast collision detection methods [10, 14] between different objects in the environment to compute the exact time of initial contact. An alternative method (which we use here) is to compute the spatial proximities of each of the end-effectors with the different objects in the environment. A spatial constraint is recognized when the objects first come in contact with each other (in the collision detection method) or when the computed proximal distance is less than some pre-specified value, ϵ. It would be possible, though computationally inefficient, to compute these collisions or proximities at every frame of the animation. In this section, we describe the various computational simplifications that we use while still being able to derive all the necessary information to recognize the spatial constraints.

3.1 Zero-Crossing

In computer vision, zero-crossings of the second derivative are commonly used for edge detection [16] in static images. For example, the Marr-Hildreth operator uses the zero-crossings of the Laplacian of the Gaussian and the Canny operator uses the zero-crossings of the second directional derivative.

In motion analysis, we can use the zero-crossings of the second derivative of the motion data to detect significant changes in the motion. The zero-crossing point in a trajectory implies changes in motion such as starting from rest, coming to a stop, or changing the velocity direction. These events were noted to have descriptive significance in [4]. When the zero-crossing point also coincides with an end-effector being proximal to another object, it implies contact of the end-effector with the object. In motion studies, this further implies that the primary agent came in contact with the object and suggests creating a spatial constraint to mark this occurrence. We record the corresponding global location of the sensor and mark it as a constraint point for the corresponding end-effector of a secondary agent. The zero-crossings enable us to compute the proximities only at *possibly relevant* frames.

3.2 Tracking Sensors

For an action, it is not necessary to track the zero-crossings of all the sensors on the human model. So, for each action, the user can specify the few specific sensors that need to be tracked. For example, in *drink from a mug*, only the sensor on the hand needs to be tracked for zero-crossings. In all actions, the sensor on the head is used as a tracking sensor for capturing the primary agent's attention.

3.3 Tag Objects

For an action, it is again not necessary to compute proximities of the tracking sensors with all the objects in the environment. As this entire technique is done as a post-process of the motion-capture session, the specific objects that are

involved in the action are already known. Using this knowledge, the user can specify the few objects in the environment that need to be used for computing the proximities. To use these, we introduce *tag objects*.

A *tag object* is associated with a site, figure, type and status. A 3D object in our environment can have a number of sites defined on it for various purposes. Each *tag object* has a *tag site* associated with it which is actually used for computing the proximities to the end-effectors. The tag figures refer to the 3D object (or parts of the 3D object) itself. This enables us to have tag sites which are not associated with the tag figure. For example, in the action of *touch the table*, the tag site may be a site on the sensor used to obtain the position on the table. But the tag figure would refer to the table itself. We can define *tag objects* on parts of the human model, thus allowing us to track body/self interactions. For example, in *drink from a mug*, one of the *tag objects* would refer to the mug with a tag site defined on the handle of the mug, and another *tag object* would refer to the head of the human model with the corresponding tag site defined at the mouth.

The *tag objects* may be of different types:

SELF: The tag figure is a part of the human model itself - *e.g.,* head.
FIXED: The tag figure does not move in the environment - *e.g.,* table.
MOBILE: The tag figure can be moved in the environment. *e.g.,* mug.

In examples considered here involving mobile objects, we assume that the agent interacts with them by grasping or holding them and moving them to another place. In other words, for at least part of the action, the mobile object is constrained to move with an end-effector of the agent. In such cases, the status flag of the *tag object* indicates if the tag object is CONSTRAINED to the agent or if it is FREE.

3.4 Spatial Constraint

The process of automatically recognizing a spatial constraint can be summarized as follows: For each tracking sensor, Euclidean distances are computed, at every zero-crossing frame, between the tracking sensors and each of the tag sites. If any of these distances is less than some predefined value, ϵ, a spatial constraint is said to exist between the tracking sensor and the corresponding tag object. The exact location of the spatial constraint to be used for another agent depends on the type of the tag object and its status (if it is a mobile object). This is discussed in detail in the next section. Figure 1 shows the trajectory of the hand tracking sensor for the example *touch the table* and Figure 2 shows the corresponding plots of the distance between the tracking sensor and a tag site (sensor on the table) and the zero-crossings of the accelerations. It can be clearly seen that a spatial constraint is established when the zero-crossings coincide with the close proximity of the tracking sensor and a tag object.

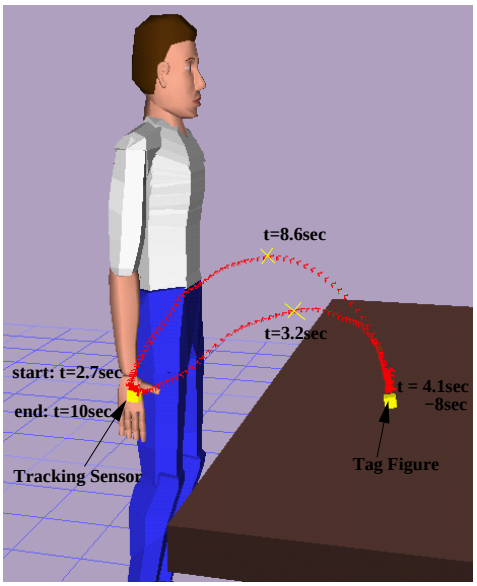

Fig. 1. Trajectory of the tracking sensor in the example *Touch the Table*

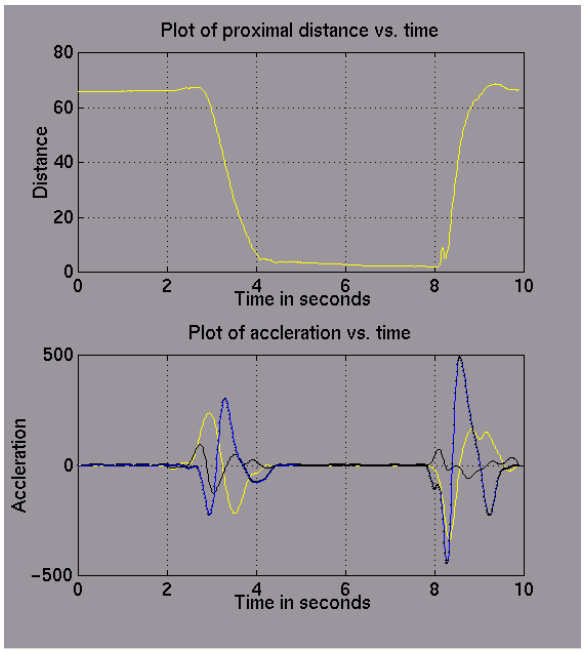

Fig. 2. Plots of spatial proximity and zero-crossings

4 Determination of Spatial Locations of Constraints

The spatial proximity of each tracking sensor from each tag site is computed at the zero-crossings of the tracking sensor. If a spatial constraint is recognized as outlined above, then the global locations of the constraint need to be used as a constraint location during the secondary agent's action. The global location of the constraint is computed based on the type of the associated *tag object*. If the *tag object* is of type FIXED or MOBILE, then it refers to an external 3D object and the absolute location of the tracking sensor is used as the location of the constraint. But, if the *tag object* is of type SELF, then the relative global location of the tracking sensor is used as the location of the constraint. The relative global location is computed by taking into account the size (lengths of the different segments) of the secondary agent. For example, in *drink from a mug*, for the first spatial constraint established during grasping the mug to pick it up, the absolute global location of the hand sensor at the time of first contact with the mug is used as the location of the constraint. For the second spatial constraint (of the same action) established during holding and bringing the mug to the mouth, the relative global location of the hand sensor when the mug comes in contact with the lips is used. This will cause the secondary agent to grasp the mug at the same location as the primary agent but will hold the mug to his mouth correctly, which may be at a different global location based on the difference in sizes between the two agents.

5 Mapping Motions to Other Agents

Once the locations of the spatial constraints are determined, a combination of different techniques may be employed to generate the movements for the secondary agent. To generate efficient motions, the optimization techniques described in [9, 13, 12] may be used. Here, we describe a simple technique to generate the motions. We decompose the joints in the human body into different kinematic joint chains (Fig. 3). We consider each joint chain separately. For the set of joints which are not contained in the same hierarchical chain as any of the tracking sensors, the joint angles computed for the primary agent may be proportionally mapped to the secondary agent. This is possible as they do not have additional constraints imposed on them. All the other joints are driven by the new spatial constraints computed above. As each joint chain is treated separately, it is very important to achieve global synchronization between the different joint chains during the entire action. To do this, we preserve the same timing information *i.e.*, the second agent takes the same amount of time as the primary agent to complete the action. In an effort to maintain the same action style (frame-wise variations in the angular velocity), we modify the speed transform method used in [1].

To solve for the new spatial constraints, a trajectory has to be traced for each joint in the chain containing the tracking sensors. For this, we first use inverse kinematics [20] to solve for the spatial constraints at each of the zero-crossing

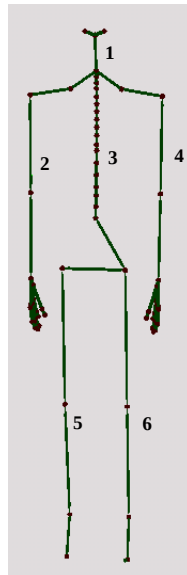

Fig. 3. Sets of joint chains defined in the human model

proximal frames. We then do a linear or spherical linear interpolation in the joint angle space for each *time period* defined between any two successive zero-crossing proximal frames. The interpolating factor $\hat{s}$ is derived by computing the normalized distance moved in the joint space by the primary agent at each frame during the corresponding *time period*:

$$s = \int_0^t |\dot{\theta}(\tau)| d\tau \tag{1}$$

where t is time, s is the angular distance moved along the trajectory, and $\dot{\theta}(t)$ is the velocity vector of the joint. The data is normalized along the trajectory:

$$\hat{s} = \frac{\int_0^t |\dot{\theta}(\tau)| d\tau}{\int_0^{t_{end}} |\dot{\theta}(\tau)| d\tau} \tag{2}$$

where t_{end} is the duration of the basic period. These interpolating values help maintain the angular velocity profile of the primary agent during the course of the action and is independent of the difference in spatial distance covered during each *time period* by the two agents.

6 Visual Attention Tracking

Capturing and maintaining visual attention is very important for movement realism in the secondary agent. Without it, actions appear unnatural even if

all the other constraints are correctly satisfied. For example, while picking up an object, the secondary agent would look extremely unnatural if she looked at some other point in space. Here, we use the above technique to easily address the visual attention constraint.

During interaction with objects, we tend to always look at the object that we are interacting with (at least when we first come in contact with it). This direction is automatically captured for the primary agent by the sensors on the head. If we naively map head motions of the primary agent to a different secondary agent, this gaze direction will be lost and *cannot be re-captured by simple signal processing techniques.* Instead, we define the sensor on the head as a *tracking sensor.* The zero-crossings in the acceleration space of the head sensor indicate a change in gaze direction or indicate gaze at a specific point in space. During these zero-crossings, we check for visual attention constraints by using the line of sight of the agent. For efficiency, we compute the intersections of the line of sight with the bounding boxes of the tag objects only. If there is any tagged object in the direction of the line of sight, the global location of the point of attention is calculated and used as the visual (alignment) constraint for the secondary agent during its motion computation. We use a head-eye tracking model to solve for the joint angles in the eyes, head, and neck at the gaze direction zero crossing frames. For the remainder of the frames, we use joint angle interpolation while maintaining the angular velocity profile as outlined above.

7 Results

We have tested this technique by mapping the actions of an adult to the virtual model of a nine year old child. We have captured *touching a table* which involves only a FIXED tag object and *drinking from a mug* which involves a MOBILE object. In both cases, we were able to successfully recognize the spatial constraints and map the motions correctly to the second agent. Of these, the example of *drink from mug* is more complicated and we discuss this in detail here.

In the example of *drink from mug*, the primary agent bends over, picks up a mug from the table, drinks from it and places it back on the table. Figure 4 shows the the plots of the trajectories of the hand end-effector of the secondary agent before and after abstraction. Before abstraction, the trajectory obtained is the result of direct mapping of the joint angles of the primary agent to the child model. It can be clearly seen that the constraint of picking the mug cannot be satisfied. But after the automatic recognition of spatial constraints and subsequent remapping of the motions as outlined in this paper, the motion of the secondary agent is corrected as can be seen by the modified trajectory.

Figure 5 shows the various stages of the drinking motion as captured for the primary agent. Figure 6 shows the various stages of the drinking motion after abstraction and mapping have been applied.

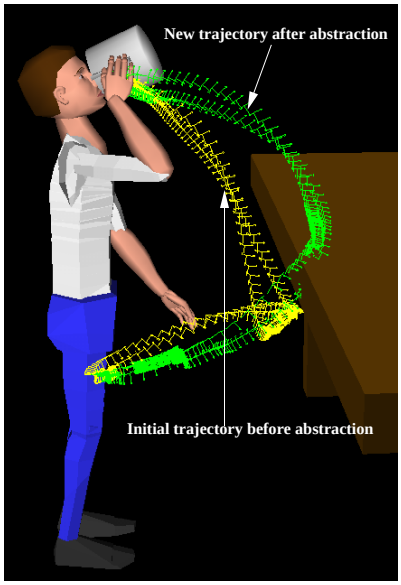

Fig. 4. Trajectory plots of the secondary agent's hand end-effector (corresponding to the tracking sensor on the hand of the adult) before and after abstraction

8 Conclusion

We have presented a new technique to automatically recognize and map spatial and visual constraints to other different sized virtual humans. This could be the basis of a very useful tool for motion capture and animation that enables automatic *semantically consistent* modification of captured data involving interactions with space and self. The potential exists for extending this technique to real-time (on-line) execution.

We have used simple interpolation techniques to generate the trajectories. We currently do not consider collisions of the new trajectory with other objects in the environment. For example, there is a possibility of collision of the secondary agent's hand with the table while reaching for the mug although there was no collision in the path of the primary agent's reach. This could be easily modified by imposing additional constraints during the trajectory generation [2].

Acknowledgments

We would like to specifically thank Harold Sun for his help in capturing all the data and Christian Vogler for building all the necessary drivers.

This research is partially supported by the following grants: Office of Naval Research K-5-55043/3916-1552793, DURIP N0001497-1-0396, and AASERTs N00014-97-1-0603 and N0014-97-1-0605, Army Research Lab HRED DAAL01-97-M-0198, DARPA SB-MDA-97-2951001, NSF IRI95-04372, NASA NRA NAG

Fig. 5. Different stages in *drink from mug* of the primary agent (adult male)

Fig. 6. Different stages in *drink from mug* of the secondary agent (a nine year old child) - after mapping

5-3990, National Institute of Standards and Technology 60 NANB6D0149 and 60 NANB7D0058, and JustSystem Japan.

References

1. Kenji Amaya, Armin Bruderlin, and Tom Calvert. Emotion from motion. In *Graphics Interface*, pages 222–229, 1996.
2. N. Badler, R. Bindiganavale, J. Granieri, S. Wei, and X. Zhao. Posture interpolation with collision avoidance. In *Proc. Computer Animation*, pages 13–20, Los Alamitos, CA,, 1994. IEEE Computer Society Press.
3. N. Badler, C. Phillips, and B. Webber. *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press, New York, NY, 1993.
4. Norman Badler. *Temporal Scene Analysis: Conceptual descriptions of object movements.* PhD thesis, CS, University of Toronto, 1975.
5. Norman Badler, Michael Hollick, and John Granieri. Real-time control of a virtual human using minimal sensors. *Presence Journal*, 2(1):82–86, 1993.
6. Joshua Bers. A body model server for human motion capture and representation. *Presence*, 5(4):381–392, 1996.
7. Armin Bruderlin and Lance Williams. Motion signal processing. In *Computer Graphics Proceedings*, pages 97–104. SIGGRAPH, 1995.
8. Sonu Chopra. Where to look? automating some visual attending behaviors of human characters. Technical Report IRCS-98-17, University of Pennsylvania, 1998.
9. Michael Cohen. Interactive spacetime control for animation. In *Computer Graphics Proceedings*, volume 26, pages 294–302. SIGGRAPH, ACM, 1992.
10. D.W. Johnson E. G. Gilbert and S.S Kerthi. A fast procedure for computing the distance between objects in three dimensional space. *IEEE Journal on Robotics and Automation*, RA-4:193–203, 1988.
11. Michael Gleicher. Motion editing with spacetime constraints. In *Symposium on Interactive 3D Graphics*, pages 139–148. ACM, ACM, 1997.
12. Michael Gleicher. Retargetting motion to new characters. In *Computer Graphics Proceedings*, pages 33–42. SIGGRAPH, ACM, 1998.
13. Michael Gleicher and Peter Litwinowicz. Constraint-based motion adaptation. *The Journal of Visualization and Computer Animation*, 9(2):65–94, 1998.
14. Thomas C. Hudson, Ming C. Lin, Jonathan Cohen, Stefan Gottschalk, and Dinesh Manocha. V-collide: Accelerated collision detection for vrml. In *Proceedings of VRML*, 1997.
15. Tom Molet, Ronan Boulic, and Daniel Thalmann. A real time anatomical converter for human motion capture. In *Eurographics Workshop on Computer Animation and Simulation*, pages 79–94, 1996.
16. Vishvjit S. Nalwa. *A Guided Tour of Computer Vision.* Addison-Wesley Publishing Company, 1993.
17. Charles Rose, Brian Guenter, Bobby Bodenheimer, and Michael Cohen. Efficient generation of motion transitions using spacetime constraints. In *Computer Graphics Proceedings*, volume 30, pages 147–154. SIGGRAPH, ACM, 1996.
18. Sudhanshu Semwal, Ron Hightower, and Sharon Stansfield. Closed form and geometric algorithms for real-time control of an avatar. In *Proceedings of VRAIS*, pages 177–184, 1996.

19. Sudhanshu Semwal, Ron Hightower, and Sharon Stansfield. Mapping algorithms for real_time control of an avatar using eight sensors. *Presence*, 7(1):1–21, February 1998.
20. Deepak Tolani. *An inverse kinematics toolkit for human modeling and simulation.* PhD thesis, CS, University of Pennsylvania, 1998. In preparation.
21. Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Computer Graphics Proceedings*, pages 91–96. SIGGRAPH, ACM, 1995.
22. Douglas Wiley and James Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, pages 39–45, November/December 1997.
23. Andrew Witkin and Zoran Popovic. Motion warping. In *Computer Graphics Proceedings*, pages 105–108. SIGGRAPH, ACM, 1995.
24. Jianmin Zhao and Norman Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

This article was processed using the LaTeX macro package with LLNCS style