SAND96-0694C
CONF-9606134--1

Techniques for Active Embodiment of Participants in Virtual Environments*

Ron Hightower
Sharon Stansfield
Sandia National Laboratories

RECEIVED
MAY 15 1996
OSTI

Abstract

This paper presents preliminary work in the development of an avatar driver. An avatar is the graphical embodiment of a user in a virtual world. In applications such as small team, close quarters training and mission planning and rehearsal, it is important that the user's avatar reproduce his or her motions naturally and with high fidelity. This paper presents a set of special purpose algorithms for driving the motion of the an avatar with minimal information about the posture and position of the user. These algorithms utilize information about natural human motion and posture to produce solutions quickly and accurately without the need for complex general-purpose kinematics algorithms. Several examples illustrating the successful application of these techniques are included.

Introduction

The requirements for representing participants within virtual environments (VEs) are highly dependent upon the application area. In the majority of current Virtual Reality (VR) systems, the user is not represented at all. Architectural walk-through or scientific visualization, for example, require that the user see, but not that she/he be seen. If the user is allowed to manipulate aspects of the virtual environment, for example open doors or pick up virtual objects, then she/he is provided with a pointer or disembodied hand to carry out such functions. In more complex simulation environments, such as those used by the military, active participants are usually represented by a graphical model of the vehicle which they are operating. So, for example, SIMNET is populated by air and ground vehicles representing and controlled by users. More recently, applications such as dismounted infantry[1] and teleconferencing[2] have begun to represent users as rudimentary human-like icons. This provides multiple participants with some sense of who else is present in their virtual world. These representations are usually simple and quite limited in their capability to represent a user's body language and actions.

Current research at Sandia National Laboratories is exploring the use of VR for situational training and mission planning and rehearsal of small team, close quarters operations. Areas requiring such operations include security and law enforcement (hostage rescue), military (urban combat) and operations other than war. These applications require that participants be represented within the VE with a much higher fidelity than do those discussed above. It is important that team members be able to see each other as full human figures. Position, posture, gesture and body language are all vital components of team coordination and communication. When the mission is a covert operation, these become even more important, as verbal communication is kept to a minimum. Therefore, if VR is to be used to plan and rehearse such missions, the virtual representations of team members must be capable of reproducing motions and actions with reasonable fidelity. In addition, VR imposes several additional requirements. First, and most important, the behavior of the participant's virtual self, which we call his/her Avatar, must be updated at near real-time to reflect the immediate actions of the user. In addition, the number of sensors/trackers used to obtain information concerning the participant should be small to minimize the amount of data which must be collected and processed, and also to avoid encumbering the user.

This paper presents work and preliminary results in the development of techniques for actively embodying a participant within a VE. The focus of the work is two-fold. First, it is desired to have an Avatar reproduce the actions of a user with enough fidelity to satisfy the close quarters training application discussed above. Second, the simulations used for this training will also

* Presented at the 1996 **IMAGE Conference** Scottsdale AZ 23-28 June 1996.

be populated by virtual actors -- purely computer-generated humans. The techniques developed here will also be applied to creating realistic motions and actions for these virtual humans.

Related Work

There are many different components to the problem of representing human figures and their motions in virtual environments. Solutions range from traditional animation to capture and use of motions performed by real people. The entertainment industry, for example, has successfully used the latter technique to create highly realistic actions for virtual humans. Generally, a large number of landmarks on a person's body are tracked using video-based devices. Proprietary algorithms are then used to perform kinematic analysis and to create motions for the virtual actors. Unfortunately, while these techniques produce very realistic motions, they have two drawbacks -- the systems used are generally expensive and the virtual character motion is not generated real-time. As a result, these techniques do not lend themselves to virtual reality applications, where it is necessary for the position and posture of the virtual actor to keep pace with that of the user. Another approach to creating virtual humans is to model the human figure as a set of joints and to use motion algorithms to create motions specified by end, or goal, postures. Examples of such methods are 3-D keyframing[3], kinematics[4], dynamics[5], joint-dependent deformations[6], stochastic noise functions[7], and combinations of the above[8] Many of these techniques are better suited to generating motions for virtual humans than to recreating realistic motions of actual people. Kinematics has been applied most successfully to this latter problem. Here, the goal is to make the kinematic model simple enough to allow solutions to be generated in real-time, while still ensuring that these motions will be a reasonable representation of the user's actions. If the model is too simple, then the motions will appear robot-like. If it is too complex, then multiple solutions may be generated, only one of which will correctly correspond to the user's motion. One way to overcome this problem is to attach more sensors to the user, thus adding more constraints to the system. This solution, however, can be expensive, encumbering, and slow. Another approach, which we address here, is to simplify the model and to use knowledge of natural human motion to generate the most appropriate solution.

The Avatar Driver

In this section, we present algorithms for driving the motions of the avatar based upon the actions of the user to whom it is slaved. The basic problem of the avatar driver is one of inverse kinematics -- using data about the position and orientation of sites on the user's body to determine the joint angles of the user's graphical representation, or avatar. If every independent body part has a unique sensor, the avatar driver can simply echo the position of each body part. Reducing the number of sensors makes the task more difficult, as it results in an under-specified avatar. In our current setup we have limited ourselves to four sensors. These sensors are mounted on the user's head, lower back (pelvis sensor) and hands. Each sensor provides information about its position and orientation (a frame). From these four frames the avatar driver must compute 69 joint frames in order to position the avatar. Because the avatar in this case is under-specified, many of the joint angles must be determined based on information other than the sensor data. Knowledge of human movement, for example, is used to position the legs in a natural stance in the absence of direct sensor data -- the pelvis sensor is used to constrain the range of possible positions. The shoulders, which also lack direct sensor tracking, are also positioned using such heuristics.

Our original avatar driver used a powerful, general purpose algorithm for solving inverse kinematics [9]. Given a target frame for a part of the body, this software would find appropriate joint transforms to realize the target position. The algorithm used information about joint constraints, but had little or no knowledge about comfortable or common limb positions/postures. As a result, when multiple solutions were possible, the algorithm would often "select" a solution which was not representative of the posture of the user. In addition, because of the general-purpose nature of this algorithm, the time required for calculating a solution was often long enough to cause delays in the updating of the avatar's position. This would cause the motions of the avatar to fall behind those of the user, making it difficult for him/her to maintain a sense of presence within the virtual world. To overcome some of these problems, we have begun developing a special-purpose avatar driver that will allow us both to incorporate knowledge about natural body postures and to explore the use of different paradigms for body positioning. Incorporating knowledge about postures will enable the avatar driver to create realistic human postures given only sparse sensor data. The current avatar driver replaces the single, general purpose method mentioned above with a handful of special purpose algorithms for inverse kinematics. In the following section, we will discuss the algorithms for positioning the head and shoulders; the hands and arms; and for moving from a standing to a kneeling posture in the absence of direct sensor input.

Geometry of the Avatar

We are currently (and temporarily) using human figure geometry derived from the Jack® software developed at the University of Pennsylvania[10]. This figure hierarchy has 68 body segments (most are in the hands and the upper torso: 32 in the hands and 17 in the spine.) Each body segment is represented by a joint frame that must be computed each time the avatar is repositioned.

The root of the hierarchy is located in the pelvis segment, and all other parts are in kinematic chains attached to that root. Sensor *S1*, worn in the small of the back, corresponds closely to the root of the avatar. The frame data from this sensor can be used directly to position and orient the figure. However, in typical usage the data from the pelvis sensor is combined with other information before being used to position the avatar relative to the virtual environment of which it is part.

Head and Shoulders

The algorithm for positioning the head and shoulders of the avatar is currently an ad hoc approximation. Over a small range of movement the head bends and twists correctly, but with larger movements gross discrepancies appear. The current method is associated with the spinal structure of the current avatar hierarchy, which we plan to redesign in the near future. For this reason, we have not attempted to correct the current algorithm for positioning the head and the upper torso over larger ranges of motion. Briefly, the head is positioned in the following fashion. The location of the head (associated with sensor *S0*, which is attached to the headmounted display worn by the user) is determined with respect to the pelvis frame. The bend and twist of this relative frame is distributed evenly among the seventeen spinal segments of the upper torso. Because the spine of the avatar hierarchy reflects the double curvature of the real human spine, this simple distribution of angular displacement is not very accurate. It has, however, been sufficient for current needs.

Hands and Arms

Once the position of the upper torso has been determined, it is a straightforward process to accurately place the hands of the figure in the positions specified by the hand sensors (*S2* and *S3*, worn on the back of the user's hands.) The position of the elbow, however, is under-specified, since only the shoulder frame and the hand frame are known. This remains true even when the angular constraints of the wrist and shoulder joints are taken into consideration. Four factors must be considered in determining elbow position. These are joint constraints, collision with upper torso, avoidance of unnatural postures, and continuous solutions across continuous ranges of sensor data. Only the last two issues are addressed in the solution presented below.

The primary purpose of this computation is to place the wrist frame in the correct position (as specified by the hand sensor). If the arm was a single rigid link, of the same length as the distance from shoulder to wrist, then positioning that link could be reduced to specifying angles for azimuth and elevation.

Let w_x, w_y, and w_z denote the component distances from the shoulder to the wrist, relative to the shoulder frame (x axis forward, y axis points right, and z is down along the limb). Let the elevation angle be given by arccos (w_x, w_z) and the azimuth angle be arccos $(-w_y, d_{xz})$ where $d_{xz} = \sqrt{w_x^2+w_z^2}$.

In normal usage ``elevation" gives angular distance above the horizon and ``azimuth" is similar to a compass heading. Here we modify the normal orientation of these angles to create a more continuous solution. Elevation gives the arm's angular displacement away from the body, left or right, and ``azimuth" gives rotation around the y axis passing through the shoulder.

These angles would correctly position the arm if it were a rigid link of the correct length. The desired length, D, is given by

$$D = \sqrt{w_x^2+w_y^2+w_z^2}$$

We can create a rigid link of the desired length by bending the elbow an appropriate amount. Figure 1 shows the triangle formed by the upper arm, the lower arm and a segment of length D. Using the relevant identity,

$c^2 = a^2 + b^2 - 2ab\cos c$

we find these angles:

$$elbow\ angle = \pi - \arccos\frac{u^2+l^2-D^2}{2ul}$$

$$shoulder\ angle = -\arccos\frac{u^2+D^2-l^2}{2uD}$$

These two rotations force the distant from shoulder to wrist to be the desired D. A compensating rotation is applied to the shoulder so the wrist does not experience a net rotation, and only moves closer to the shoulder. If the length of the upper arm, *u,* were equal to the length of the lower arm, *l,* then the compensating shoulder rotation would be exactly one half the elbow rotation angle.

The final position of the elbow can be adjusted by rotating the arm about the shoulder's z axis, after bending the arm but before applying the elevation and azimuth rotations. Currently we apply a constant 45 degree rotation, bringing the elbow out away from the body. We plan to use this particular rotation to improve the natural look of the arm position and also to reduce collisions with the upper torso.

Once the shoulder and elbow joints have been properly rotated to bring the wrist into the correct position, a final transform must be applied to give the avatar's hand the same orientation as the hand sensor. The sensor frame, with respect to the universal coordinate system is denoted ${}^{U}_{H}T$ and the frame of the wrist, incorporating the rotations of elbow and shoulder, is ${}^{U}_{W}T$ .We want to find a transform from wrist to hand, ${}^{W}_{H}T$, which we can get by solving:

$${}^{W}_{H}T\,{}^{U}_{W}T = {}^{U}_{H}T$$

$${}^{W}_{H}T = {}^{U}_{H}T({}^{U}_{W}T)^{-1}$$

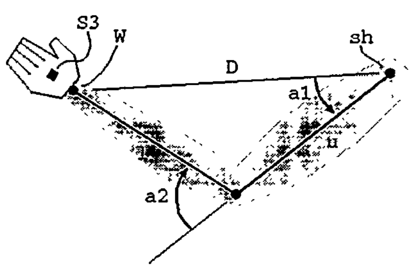

Figure 1 Bending the elbow. D distance from wrist to shoulder. *u*: length of upper arm, *l*: length of lower arm. *S3*: sensor frame, W: wrist frame, sh: shoulder frame, a2: joint angle for elbow, a1: compensating angle for shoulder joint.

Kneeling

The kneeling algorithm was created specifically for use with a medical training simulation, where the avatar has to kneel beside a virtual patient. This type of kneeling is different from the kneeling position used for firing a weapon, but the two are indistinguishable in terms of sensor data. In order to incorporate both kneeling behaviors into the same avatar driver it would be necessary for the system itself to choose between kneeling postures based on the environmental context, such as the proximity of a virtual casualty or the drawing of a weapon.

The kneeling behavior for the medical simulation is a very constrained set of motions. The assumption is that the user will not move horizontally once the knees have made contact with the ground. This prevents intersection between the avatar knees and the virtual patient. Unfortunately people do move around on their knees, which can lead to a discrepancy between user and avatar position. Once we have implemented full collision detection we can allow the avatar to safely move while kneeling near the virtual patient.

Figure 2 shows the two states of the kneeling process. When the pelvis sensor drops below a certain height value the avatar enters the knee bending state. Upon entering this state the position of the toe is recorded, and the toe remains fixed in this location until the avatar returns to the full standing position. During the knee bending state the position of the legs is controlled by the distance from pelvis sensor to the ground (horizontal displacement is ignored). The ankle is fixed at a 90 degree angle. Using the same triangle identity as before we find the three angles for the leg.

$$knee\ angle = \pi - \arccos\frac{u^2+l^2-dgl^2}{2ul}$$

$$hip\ angle = -\arccos\frac{u^2+dgl^2-l^2}{2udgl}$$

$$toe\ angle = -\arccos\frac{l^2+dgl^2-u^2}{2ldgl}$$

When the knee bending state is entered the frame of the toe, B_0, is recorded. The toe frame, B is computed by traversing the hierarchy from the root frame through the hip, knee, ankle, and toe rotations: $B = R_{toe} R_{ankle} R_{knee} R_{hip} Root$. Combining the rotations together gives $B = R_C Root$. During the knee bending state we want to force the toe to be at the original toe frame B_0, by adjusting the root frame. Hence, for $B = R_C Root$ we want to find $Root'$ so that $B_0 = R_C Root'$. Solving for R_C and substituting gives $Root' = Root B^{-1} B_0$. Once the joint angles have been computed, according to the height of the pelvis sensor, an initial toe frame is found. This toe frame is then used to find a new root matrix which guarantees the toe will occupy the desired position. If, during the knee bending state, the pelvis sensor drops sufficiently low then the avatar's knee will come in contact with the ground plane. At this time the avatar enters the knee-on-ground state. In this state the lower legs remain fixed in place, and the upper leg is only allowed to rotate about the knee joint. The angle of rotation (the value K in Figure 2) is determined by the horizontal displacement of the pelvis sensor and is given by $a_4 = \arcsin(K/u)$. The rotation of the knee joint is increased by a_4 and in compensation the rotation of the hip joint is decreased by a_4. The root matrix is recomputed using $Root' = Root B^{-1} B_0$ as before.

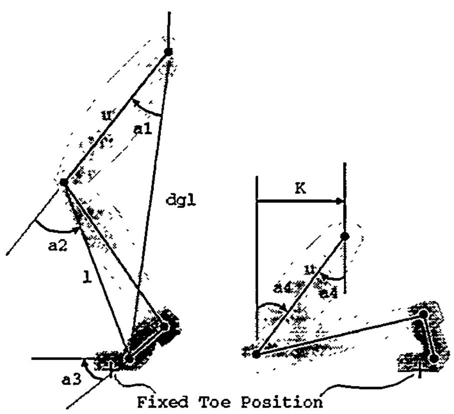

Figure 2 Bending the knees. dgl: distance from ground level, u: length of upper leg, l: length of lower leg. a2: joint angle for knee, a1: compensating angle for hip joint, a3: compensating angle for toe joint. Kneeling. K: horizontal displacement of pelvis, a2: additional angular displacement for knee and hip compensation.

Preliminary Results

Figures 3 - 7 illustrate the effectiveness of the techniques presented in this paper. Each figure shows the user in a different pose and the respective position of the avatar as determined by the avatar driver.

Acknowledgments

This work was performed at Sandia National Laboratories and was supported by the US Department of Energy under Contract DE-ACO4-94AL85000. The authors would like to thank Dan Shawver and Sudhanshu Semwal for their input and contributions.

References

1. Granieri, J., Crabtree, J. and Badler, N., "Production and playback of human figure motion for 3D virtual environments," *Proceedings of the IEEE Virtual Reality Annual International Symposium,* Research Triangle Park, NC, March 1995.

2. "Sharing cyberspace made easier," *CyberEdge Journal,* 5(4), Nov/Dec 1995.

3. Calvert, T., "Composition of realistic animation sequences for multiple human figures," *Making Them Move: Mechanics, Control and Animation of Articulated Figures,* Badler, Barsky, and Zeltzer, Eds, Morgan Kaufman, 1991.

4. Badler, N., Phillips, C. and Webber, B., *Simulating Humans: Computer Graphics, Animation and Control,* Oxford University Press, 1993.

5. Armstrong, W. and Green, M., "The dynamics of articulated rigid bodies for purpose of animation," *The Visual Computer*, 1(4), 1985.

6. Magnenat-Thalmann, N. and Thalmann, D., "Human body deformations using joint-dependent local operators and finite-element theory," in *Making Them Move: Mechanics, Control and Animation of Articulated Figures,*

Badler, Barsky, and Zeltzer, Eds, Morgan Kaufman, 1991

7. Perlin, K., "Real time responsive animation with personality," *IEEE Transactions on Visualization and Computer Graphics,* 1(1), March 1995.

8. Singh, K., Ohya, J. and Parent, R., "Human figure synthesis and animation for virtual space teleconferencing," *Proceedings of the IEEE Virtual Reality Annual International Symposium,* Research Triangle Park, NC, March 1995.

9. Badler, N., Hollick, M. and Granieri, J., "Real-time Control of a Virtual Human using Minimal Sensors," *PRESENCE,* 2(1), 1993.

10. *Jack 5.0 User's Guide,* Center for Human Modeling and Simulation, University of Pennsylvania, Philadelphia, PA, 1992.

Figure 3

Figure 4

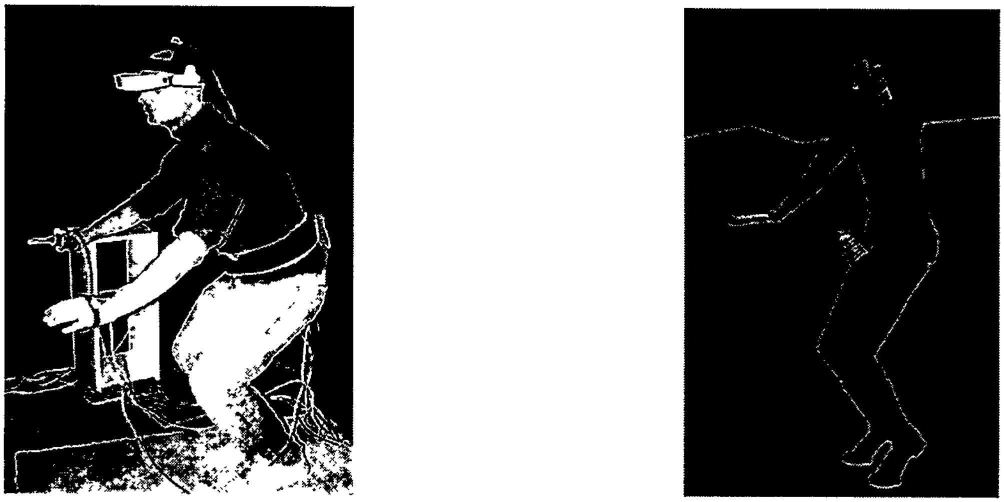

Figure 5

Figure 6

Figure 7

DISCLAIMER