# Animation Support for Networked Virtual Environments

Hyeongseok Ko
School of Electrical Engineering
Seoul National University
ko@kdb.snu.ac.kr

Abstract: This paper presents animation techniques and issues involved in virtual environments where the participants interact with each other through a network. The state of the participant should be recognized at each local site, and broadcast to the other sites. Because information exchange is minimal, animation techniques are applied to convert the incoming low DOF parameters into high DOF joint angles that completely determine the configuration of the agents at each frame. As a case study, a software system VRLOCO is introduced, which has been developed by the author over the last five years. From a simple stream of body center positions, VRLOCO generates realistic curved path human locomotion in real-time. Based on the heading direction and speed, the locomotion automatically switches among five different primitives: walking, running, lateral stepping, backward stepping, and turnaround. The techniques presented here proved robust and faithful: the algorithm is not sensitive to the noise in the data, and the resulting animation conforms well with the original data.

## 1 Introduction

*A soldier is sitting on a VR input device that detects his upper and lower body motion, and broadcasts the result to other simulation sites at 30Hz. He is surrounded by three big screens located in the front, left, and right, so that they cover 270° field of view. The screen projects the battle field in which other blue and red forces interact each other. The actual people controlling the soldiers' behavior are scattered all over the world.*

The above is so-called the Individual Soldier Mobility Simulator[1] [5], which is a currently on-going project in U.S. military, to create a virtual combat environment and train infantry soldiers.

[1] The Individual Soldier Mobility Simulator is compatible with the DIS (Distributed Interactive Simulation [1]) protocol.

In such a simulation, the interactions among the agents become an important part. This paper is about how to provide the visuals for interactive simulations.

In order to describe the problem more clearly, let's assume the system architecture depicted in Figure 1 throughout this paper. It shows the information flow among different modules in a networked virtual environment. VR input devices detect the movement of the participant, and broadcast this information to the other sites. The viewing parameters are extracted from the head position/orientation, and used in rendering the scene at the local participant's view. The animation module collects the information about the other agents, and determines the detailed configuration, and sends it to the rendering module for display.

First, we should decide what kind of information should be exchanged though the network. Transmitting the images of the agents wouldn't work: not only it will congest the traffic, but the image is at a specific view and will not give the right view to other users. In most implementations, a set of parameters (such as position, velocity, etc.) is chosen that represents the current state of the agent. VR input devices are attached to the participants to detect these parameter values during the simulation. The parameters are broadcast to the other sites to notify the the agent's current state.

At each site, those parameters should be collected to recognize the status of the other agents, and the scene should be rendered at the local participant's view. The information exchange is usually minimal, and we cannot simply reconstruct all the details of the agents from the incoming parameter values. This calls for the animation techniques that produce the complete high DOF configuration from the crude information.[2]

This paper discusses the animation techniques applicable to the above problem of reconstructing the detailed motion from simple motion parameters. Also, as a case study, VRLOCO will be introduced. It is a software system developed by the author over the last five years. VRLOCO generates realistic human locomotion animation from simple positional input streams.

[2]Of course, if we collect and exchange exhaustive data, animation will be a simple process. Actually, we can attach virtually unlimited number of sensors (e.g., potentiometers). However, each sensor has a limited accuracy. Moreover, it is difficult to mount a sensor firmly on human flesh so that it measures accurate joint angle or position. The errors introduced from each individual components can result in a shaky, unrealistic overall motion. If the sensors need to be connected with wires, the participant's movement will be constrained by so many attachments. Also, the network traffic will increase significantly. This can be a severe problem if a large number of (say, hundreds of thousands) agents are participating in the simulation.

## 2 Animation Techniques for Networked VEs

Many fancy rendering features are supported from the graphics system venders these days. However, animation is still the application programmer's burden. Even though we can render a quite realistic tennis court and players, it is still very difficult to produce realistic animation of tennis match. As the rendering techniques provides a fairly realistic scene, people expect to see realistic motion and behaviors. Thus the bottleneck in computer simulation is now at the animation techniques.

The role of our animation technique is to convert a set of low DOF parameters (let's call it the *commanded input*) into a another set of high DOF parameters (let's call it the *realized motion*) that conforms with the given low DOFs and generates realistic motion as a whole.

### 2.1 Realism vs. Accuracy

Obviously, the above is a redundant process. i.e., there are an infinite number of solutions that realize the given commanded input. The choice among these multiple solutions should be based on how realistic the motion is.

Frequently, the data contains some noise. In this case, we should isolate the noise rather than producing unnecessarily faithful realized motion by following the random fluctuations in the data.

Even without the noise, the realism of the motion can be enhanced if we allow some flexibility in satisfying the commanded input constraints. For example, in animating human locomotion on a flat level surface, let's suppose a stream of the body center position (BCP[3]) is the commanded input. A sudden change in BCP direction or speed can be accommodated into a smooth realistic motion if several parameter values are delayed, ignored, or averaged.

[3]BCP is the center of the pelvis, projected onto the ground.

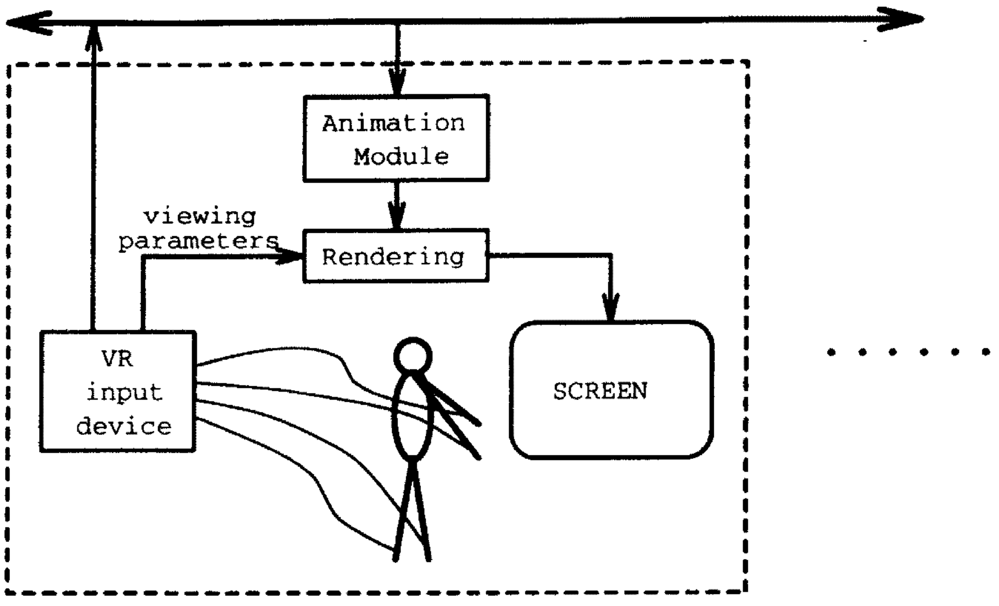

Figure 1: The System Architecture of a Networked Virtual Environment

There might be a tradeoff between realism and precise following of the BCP. Considering the fact that many systems that do not care locomotion at all (e.g., the systems that uses sliding ghosts), achieving the BCP is the primary thing and the realism of the motion is the secondary thing. Ghosts that move accurately would be more useful rather than realistic human moving randomly.

In DIS simulation [1], the accurate positioning of the soldiers in the scene is very important for the bullet trajectory analysis, and deciding who is shot. If a soldier aimed accurately and shoot an enemy by looking at the screen, but if there is a large discrepancy between where the enemy is actually located and where the enemy looks located, the participants will get confused at the results.

On the other hand, small discrepancies are permissible if the result of the simulation is not affected and these discrepancies improve the realism. Generally, the realism of the animation increases as we allow larger discrepancies. However, each application needs to have its own level of accuracy. Therefore we put the *grace error bound*, $\epsilon$, that compromises the realism and accuracy. As long as the error between the commanded input and the position of the animated figure (in the realized motion) stays within $\epsilon$, the validity of the simulation and realism of the motion are guaranteed.

Clearly, both realism and accuracy should be considered for the measure of success. The grace error bound provides an effective, predictable mechanism to combine the two conflicting aspects necessary for a successful animation of the commanded input parameters.

## 2.2 Using the Context Window for the Missing DOFs

Now, let's look at the problem of obtaining the missing DOFs. Even though the realized motion consists of hundreds of joint angles and global position, once we know it is coming from a certain pattern of motion, we might be able to guess those detailed parameters that conform with the commanded input sequence. Note that the mapping between the commanded input and realized motion depends on the *context*. For example, if the current context is slow speed walking, then the BCP should be interpreted accordingly to generate all the joint angles.

The next question is how to find out the current context from the on-line commanded input sequence. The current context can be figured out through the *history analysis*. For example, if the BCP samples start to advance at a higher speed recently (during the last 0.5 second), and cannot catch them up by walking, then the locomotion may have to switch from walking to running. Thus the BCPs should be interpreted differently to generate running poses rather than walking poses.

In VRLOCO, the history analysis consists of a numerical integration performed on the BCP samples to guess the current context (i.e., the locomotion mode, which switches among walking, running, turnaround, lateral stepping, and backward stepping) [4]. Here the integration interval is called the *context window* of the history analysis.

A non-zero context window may introduce a latency in the context switching. Here the latency is the delay between the real world context switching and the context switching in the virtual environment. For example, if the BCP starts to advance faster, the walking switches to running (animation in the virtual environment) a few frames later, after the history analysis concludes the BCP pattern indeed carries running motion rather than walking.

As we use a bigger context window, the context prediction is more accurate and less sensitive to the noise. However, at the boundary of the two contexts, the duration while the samples are misinterpreted (according to the old context, which is wrong) gets longer. On the contrary, if use use a shorter context window, the context prediction may be less accurate. But if the guess is correct, the duration while the samples are misinterpreted gets shorter. Therefore increasing the size of context window does not always produce a good nor bad result. In VRLOCO (Section 3) implementation, we used the context window of ten frames, which was determined through experiments.

## 3 Case Study - VRLOCO

Virtual reality applications, especially in entertainment and training, require environments populated with multiple interacting humans. Whether the virtual humans are controlled by real people or by computer programs, a large portion of their activity will involve locomotion. VRLOCO is a "locomotion engine" designed to meet the locomotion requirements of virtual environments.

Locomotion techniques for VE applications must be:

- **realistic** - if the user sees an agent walking in a virtual environment, the realism of the motion is not less important than the realism of the environment to give the feeling of immersion.
- **broadly capable** - they should include a variety of walking modes and styles, and should not be limited to periodic stepping along simple paths on flat terrain.
- **easily controlled** - they should provide a clear high-level interface that allows control of locomotion through small number of intuitive parameters.
- **responsive** - they must be able to generate animations on-line, in continuous minimal-latency response to the control inputs.

VRLOCO combines broad locomotion capabilities — five basic locomotion primitives (walking, running, lateral stepping, backward stepping, and turnaround) plus techniques for smoothly blending between primitives — with a control interface based on simple positional streams. Because it is also fast enough to generate locomotion at greater than 30Hz, VRLOCO is well suited for use in virtual environment applications involving multiple interacting humans. Most previous approaches to human locomotion animation were not suited for VEs because of very limited locomotion capability, e.g. forward walking only, difficulty of control, and/or computational expense.

Given realistic input, VRLOCO generates realistic animation of walking, running, turnaround, lateral, and backward steps. Even for noisy and rough input the algorithm provides smooth and faithful animation, never deviating far from commanded positional goals.

VRLOCO is designed to provide "plug-in" locomotion capabilities to any applications involving multiple interacting humans. The effectiveness of our approach has been tested using several locomotion controllers — programs representing autonomous agents, standard mouse/window graphical user interfaces, and a VR input device consisting of a stationary bicycle fitted with optical encoders and a microcontroller. More details of VRLOCO can be found in [4, 3, 2].

## 4 Conclusion

In this paper, we looked at the problem of generating visuals in a VR simulation, where multiple agents are interacting through the network. The amount of information flow is limited from the network bandwidth and motion capturing techniques. The techniques presented here provides a useful guideline whenever a complicated motion should be derived from a simple set of motion parameters. Experiments using VRLOCO produces a robust and faithful result. i.e., the algorithm is not sensitive to the noise in the data, and the resulting animation conforms well with the original data [4].

## References

[1] Institute for Simulation and Training. Protocol data units for entity information and entity interaction in a distributed interactive simulation (draft), version 2.0.3. Technical Report IST-PD-90-2, Institute for Simulation and Training, Orlando, Florida, May 1993.

[2] Hyeongseok Ko. *Kinematic and Dynamic Techniques for Analyzing, Predicting, and Animating Human Locomotion*. PhD thesis, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, PA 19104-6389, May 1994. MS-CIS-94-31.

[3] Hyeongseok Ko and Norman I. Badler. Animating human locomotion with inverse dynamics. *IEEE Computer Graphics and Applications*, 16(2), March 1996.

[4] Hyeongseok Ko and James Cremer. VRLOCO: Real-time human locomotion from positional streams. *PRESENCE: Teleoperators and Virtual Environments, Special Issue on the Human Figure in Virtual Environment Systems*, 5(3), Summer 1996.

[5] David R. Pratt, Paul T. Barham, John Locke, SARCOS Inc; Micheal Hollick Micheal J. Zyda, Naval Postgraduate School; Bryant Eastman, John Granieri, Hyeongseok Ko, and University of Pennsylvania Norman I. Badler. Insertion of an articulated human into a networked virtual environment. In *Distributed Interactive Simulation Environments Conference*, University of Florida, December 1994.