ViPER: Virtual Reality Pose Estimator and Viewer

Edwin Pan
Stanford University
edwinpan@stanford.edu

Manan Rai
Stanford University
mananrai@stanford.edu

1. Introduction

3D human pose tracking is difficult. Recent works in the computer vision space have made major strides in estimating 3D human motion from monocular video. The core aspect of our project is the question: can motions captured from video be used in games? To answer this question, we built a VR game in Unity that simultaneously utilizes both video-distilled motions and traditional motion capture motions. This allows us to evaluate the accuracy and usability of these motions in their native 3D environment.

1.1. Overview

The ViPER[1] game environment is composed of four different zones (Figure 1). These are outlined as follows.

- Zone 1: Start Position
- Zone 2: Motion Evaluation
- Zone 3: Agent Interaction Evaluation
- Zone 4: Goal Position

The player has at their disposal 5 different animated motions to use for navigating around the environment. Namely: w-forward, a-left, s-backward, d-right, and SPACE-jump. Note that the left and right motions only work in adjusting the trajectory. In other words, the player cannot shuffle left or shuffle right from idle. There are also animations for "idling" and "crouch" which are useful for visualization but serve no navigational purpose. These motions were obtained from clips in the Carnegie Mellon University Motion Capture Database [1].

1.1.1 Zone 1: Start Position

This is the start location, with the player facing the large ramp that leads to zone 2. Going through the wall to the right takes the player to zone 3. Since the blocks separating zones 1 and 3 are dynamic objects, the enemy agent may attack the player depending on the patrol motion.

[1]Code is available at https://github.com/edwin-pan/ee267_viper

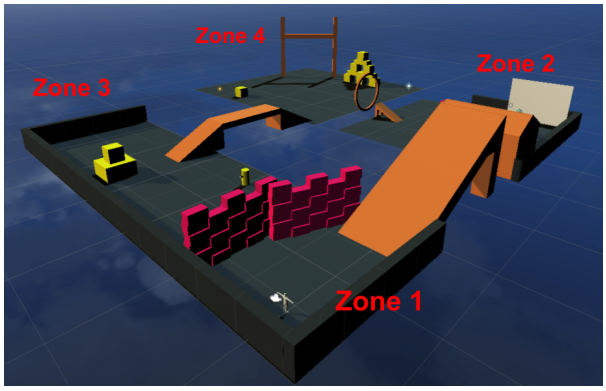

Figure 1. Overview of the ViPER environment, with zones labeled.

1.1.2 Zone 2: Motion Evaluation

This location is where the player can evaluate motions captured from video source. To generate these images, we used a pre-trained VIBE [3] motion estimator model. This model uses SMPL [4] parameters to codify human motion, and uses these parameters to regress a 3D human mesh over time. These are then converted to the FBX format and loaded into ViPER for evaluation. Within the ViPER environment, the FBX animation is played on a loop in front of the video that sourced the motion. This allows the user to navigate around the human mesh in 3D, and helps identify problems with the mesh model outputs. For our demo, we used a video of a woman doing jumping jacks to produce the motion displayed in Figure 2.

1.1.3 Zone 3: Agent Interaction Evaluation

In this location, we provide an enemy agent with a simple agitator function to help place the motion animations within a game setting. The enemy agent is governed by a state machine comprised of the states: patrol, chase, and attack [2]. When the player is outside of the range of the yellow sphere seen in Figure 3, the agent chooses a random location on the navigable surface (XZ-plane) within a predefined search radius and navigates to that region. Once the player enters the

Figure 2. Example of a motion captured from the video displayed on the plane object.

yellow region (but remains outside the red region), the agent enters the chase state. When in this state, the agent will pursue the player. When the player's location sits inside the red sphere, the agent will halt motion and open fire. The agent will fire grenades at the player, which explode after either exceeding a timer or colliding too many times.

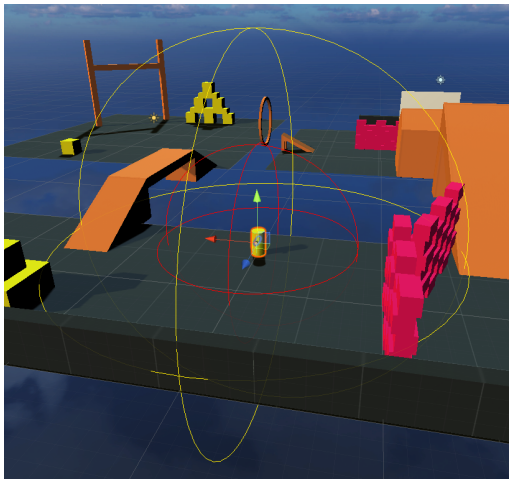

Figure 3. Enemy agent trigger zones.

1.2. Video Motions for the Player

An attempt was made to incorporate motions sourced from video data in the player controlled set of animations. This proved to untenable for a number of reasons.

First, the video sourced motions often contained temporal artifacts that made them difficult to use. For example, we had envisioned executing a dance sequence once the player reached the goal post in zone 4. However, we observed that motions containing fast non-repetitive actions often confused the model, leading to significant frame-by-frame deviations that rendered the output motions unusable.

Figure 4. Part of the cut dance sequence. Note the rapid jump in post estimation across the 0.5 second span shown.

Second, even when the models do a good job predicting the 3D motion, there are issues at the outer-most extremities of the body. Specifically, feet and hands tend to be misplaced. This is unfortunate, since these are the most expressive parts of human motion. The feet in particular do not generally respect the physics surrounding a ground plane. Rendered feet often float (as seen in Figure 4) or penetrate the ground plane (see jumping jack sequence).

Finally, reasonable 2D re-projections often exhibit hidden artifacts when viewed in 3D. For example, the jumping jack sequence tracks the overall pose well. However, viewing from the side shows that both the volume of the rendered model often changes with the motion and the motion often contains a forward lean that violates physical plausibility.

While these issues prevented us from incorporating video sourced human motion in our player animations, it's encouraging that the ViPER environment helped us identify the exact issues that need to be addressed in the 3D human motion estimation pipeline.

References

[1] CMU Graphics Lab. Carnegie Mellon University Motion Capture Database. `mocap.cs.cmu.edu`, 2001.

[2] Dave / Game Development. "full 3d enemy ai in 6 minutes! —— unity tutorial" aug. 11, 2020. [YouTube video].

[3] M. Kocabas, N. Athanasiou, and M. J. Black. Vibe: Video inference for human body pose and shape estimation, 2020.

[4] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.