Inverse Kinematics and Full-body Tracking for Virtual Reality

Initial Exploration and Experimentation

TERRELL STRONG, University of Washington

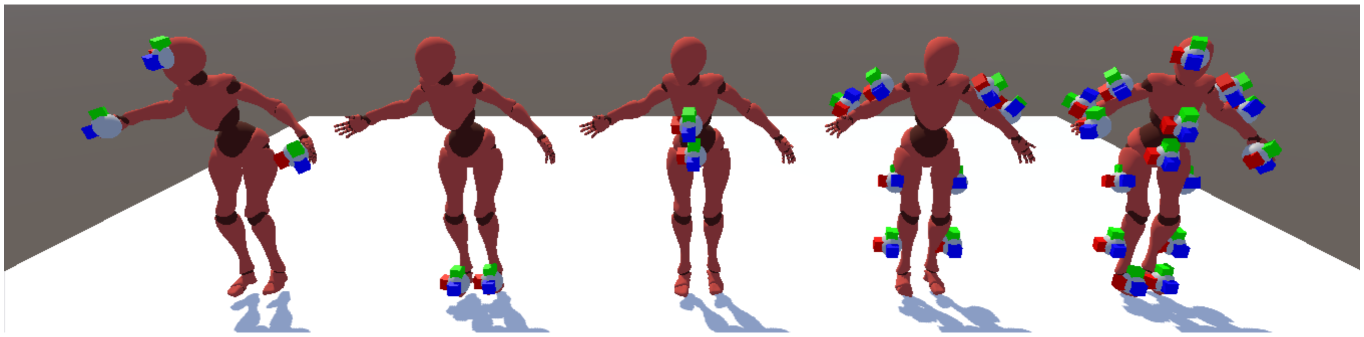

Fig. 1. This image shows how the simulated trackers were positioned and oriented on an animated character in order to create data to work from. From left to right: (1) The headset and controller tracked poses; (2) foot tracked poses, (3) chest and hips tracked poses; (4) limb tracked poses; (5) all trackers. Notably, (4) include two trackers for each limb, one above and one below each knee/elbow.

Due to limited amounts of pose data when working in VR, understanding the position of the entire body can be difficult, but making good estimations in the absence of data or using additional tracking points can improve body tracking quality and allow for heighten immersion in any given experience. Accuracy is essential if applied to a first person model in order to display a user's arms, but believably would likely be enough if trying to communicate a user's pose to someone else in VR. This project aimed to implement a basic body pose solver using Unity. To do this, tracking data was generated from an animated character so that the data provided to the solver could be either limited to traditional controller and headset tracking or expanded to include many more tracked positions on the body and limbs. Although the solver didn't reach a polished state, the work is still valuable in identifying challenges and potential future solutions.

1 INTRODUCTION

One of the most unique thing about VR is the sense of presence that it can give users by allowing them to look around and interact with virtual environments, but one of the things that I am interested in is how presence works in VR. Often times the quality of visual perception is posed as how realistic VR is, but being able the body animations of a first-person character could contribute significantly to immersion as well. Many experiences right now simply show hands as a representation of the user's virtual body because the hands and head are often the only things being tracked, but I'd like to explore how you could go about posing the rest of the body even with this limited data.

This problem is difficult because the limited data creates a lot of ambiguity, so one of the major things that will determine if a pose is good is if it is accurate. Or rather if it's inaccuracies are tolerable. I didn't address the user perception of pose accuracy, but instead used simulated data in order to allow a direct compassion between the original and generated poses.

Author's address: Terrell Strong, stront2@cs.washington.edu, University of Washington.

1.1 Contributions

While working on this project much of my focus was in creating a method of mapping general tracked data to a skeleton in order to calculate the pose of tracked body parts more accurately. A notable contribution in terms of my methodology was that I used simulated tracking points and pre-made animations in order to compare my body pose solver to a ground-truth pose. As I developed the body tracking, I experimented with heuristics to try improving the quality of the pose when limited to traditional tracking points as well. Overall, my contributions are:

- I implemented a script to generate simulated tracking points on a humanoid rig in Unity.
- I implemented the base FABRIK algorithm.
- I began implementing a body pose solver using simulated tracking points, rating pose quality by comparing it to ground-truth animations

2 RELATED WORK

In [Lang 2016], the methodology used for creating IK in the VR game *Dead and Buried* is discussed. This article was a strong motivator for me pursuing this project and relates closely in terms of what it is trying to achieve. The primary difference between my work and the article is that they worked with the constraint of using only the headset and controller positions while I am interested in how additional tracking points can assist in posing. Additionally, designing their IK solution with a specific application in mind allowed them to remove some additional ambiguity since they understood what kind of actions would be most common.

In [Aristidou and Lasenby 2011], an algorithm for IK is described that generates realistic posing solutions fairly quickly. This algorithm seems to be the go-to in many cases, so familiarizing my self with it was important. Additionally, [Aristidou et al. 2016] expands on FABRIK in order to generate poses with more complex systems of constraints. Since this was a sort of initial exploration of IK and body posing, I ended up using built in Unity functionality for much of the project and focused more on how to generate pose targets from the tracking data. I did end up re-implementing FABRIK as a means of learning how it works and how I could apply it to the character model that I was using.

In[Kim et al. 2018], the FABRIK algorithm is actually applied to arm posing in VR. This is again very closely related to my project and it's goals. I didn't try to re-implement the work done here, but instead wanted to try pushing what I could do with generated data.

3 METHOD

As mentioned earlier, the way that I went about developing the body pose solver was to get tracking data from an animated character instead of from an actual VR headset in order to allow for easier comparison to how well the generated pose fit the original pose. This meant that the first step was to first map tracking points to an animated character, then to create the solver in such a way that it could use these tracking points in place of data form actual controllers. Since I was using animations to generate pose data, it was important to consider what kind of animations to use.

3.1 Selecting Animations

The animations used were sourced from Mixamo.com, so there were a lot of choices in terms of what animations to use. To determine which animations would be most valuable in terms of developing and testing the body pose solver, I broke down what elements of the body pose would be tested by any single animations and roughly categorize animations based off of this.

- Social Gestures - Primary motion is with the arms and hands. Because of this, determining the orientation of the elbow is very important
- Focused Actions - Motions with the purpose of interacting with the world. This categorization is particularly interesting because making additional guesses and assumptions about the users intent is reasonable and can give an edge in determining how to pose the character
- Dance - These featured lots of sweeping arm and leg movements as well as significant center of mass movements. Tests with these animations can give insight into how well the hips and feet are being estimated as well as how well the limbs are rotated when being swung in an arc
- Crouch - Motions that have the character crouched or prone. These are used to test how well hip height can be estimated and how leg posing can work in cases where the player may crouch.

Comparing the importance of each category, estimating good poses for social gestures and focused actions are the most important, followed by crouching motions. The dance and challenge categories are fairly interesting because they deal a lot with how to estimate and procedurally animate locomotion, but are the lowest priority for now since lower body posing without additional tracking would be focused entirely on grounding the user believably rather than trying to imitate the ground-truth animations.

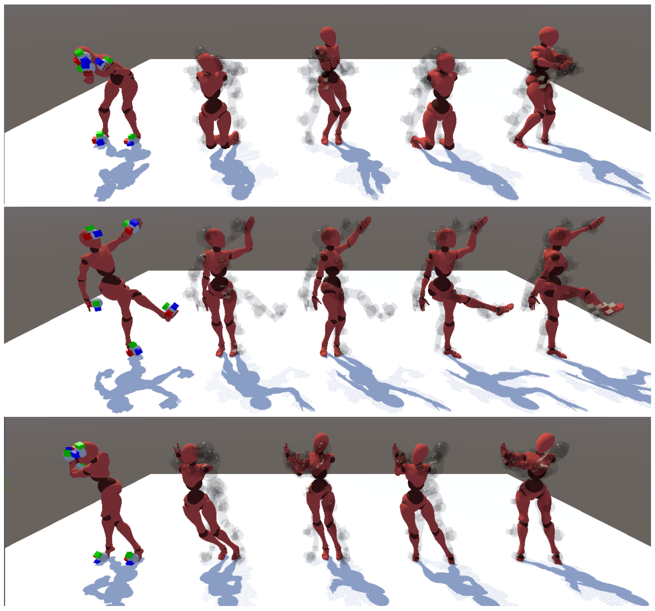
Fig. 2. three different animations that are being posed with different body parts being calculated using tracking points with the rest being estimated. In all cases, the headset and controller tracking points are being used to calculate hand and head poses. From left to right: (1) ground truth animation; (2) estimating everything; (3) calculating hips; (4) calculating feet; (5) calculating hips, feet, and elbows.

3.2 Solving Body Poses

After generating pose data, it had to be interpreted by the body pose solver in order to create and apply a pose to a new character model. The most notable thing about the pose solver is that it has to have multiple ways of determining how to pose different sections of the body dependent on what tracking data it has available to it, so naturally the solver makes separate passes over different sections of the body in order to construct a pose. The passes ended up being separated into the hands, feet, torso, and knees/elbows.

4 IMPLEMENTATION DETAILS

For this project, I worked entirely in Unity, and used the animator component to access data about the joints in the models as well as to manipulate the poses.

Testing the pose solver required two characters; the first was the ground-truth character and the second was the IK character. The pose of the ground-truth character was determined by imported animation, and the model had the generated trackers on it in order to simulate the controller and headset pose data that would be expected from VR systems, as well as additional tracking points on different segments of the body. The trackers and how they were divided can be seen in figure 1

For generating the tracker positions, I created a script that find a relevant bone in the ground-truth character, and instantiate a tracker with a predefined rotational and positional offset, parent it to the bone, and save a reference to it to be used later. It is worth noting that if the tracker positions were instead supposed to be driven by actual tracking data, a reference to the tracked object could be used instead and the offset could be calibrated in order to better reflect the user's joint positions.

After generating the tracker positions, a separate script was used to calculate apply poses to the IK character. The IK solver operated by doing an ordered pass over the spine, hands, feet, elbows, and knees. Since all trackers were generated by the previous script regardless of if they were to be used, the IK solver had parameters that determined if it would try to estimate the pose of a section based on the controller and headset trackers, or if it would calculate the pose directly using the relevant trackers. This approach of doing passes over different sections of the body was motivated by [Lang 2016].

For the hands, feet, and hips the pose was determined by essentially inverting the rotational and positional transformation from the bones to the trackers. Determining the positions of the pole vectors for the limbs is described below.

4.1 Elbow Pole Vector Positioning

In order to orient the arms, I tried using a heuristic based on the rotation of the wrist to avoid rotations that would exceed the bio mechanical limits of the wrist. Figure 3 visualizes the calculation.

The first step was to calculate the midpoint between the wrist and its pivot to get a rough estimation of where the elbow would be if the arm was fully extended. Next, the pole vector was positioned at that midpoint then offset using the wrist's local coordinate system such that it move down and outward by the estimated length of the forearm. From there, a final offset could be made to move the elbow inward or outwards using an additional parameter. The motivation of having this additional parameter was to be able to tweak the elbow position dynamically to avoid collisions with the body, but I didn't get to the point where I could drive that parameter dynamically.

4.2 Experimenting with mass

One of the heuristics that I wanted to try using for determining the behavior of the lower body was velocity, acceleration, and mass of different sections of the body. Although it wasn't integrated into the pose solver, there was some progress towards this that is worth covering. Figure 4 visualizes the following calculations.

The center of mass of the entire body is calculated as a weighted sum of all of the different centers of mass for sections of the body such as the arms, legs, head, and torso. The center of mass for each of the sections was approximated as the average position of each of its joints. For the legs, the joints used were the upper leg joint, the knee, and the ankle, and for the legs the joints used were the shoulder, the elbow, and the wrist. For the torso, all of the joints of the spine were used. For the head, the position of the head joint was taken directly. After determining the positions of each center of mass, the proportional weight of each segment in the average person was used, although these weights could be tweaked to change how the body's center of mass reacts to hand and head movements.

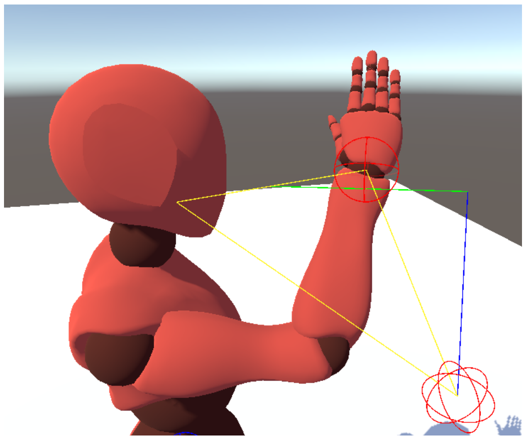

Fig. 3. Shows how the elbow pole vector is placed. The red wire sphere in the lower right represent the pole vector. The green and blue lines shows the positional offset from the wrist that is used to determine the initial elbow position. The yellow triangle shows the plane created by the initial elbow positioning, the wrist, and the estimated neck position.

Using the center of mass, the stability of the character was then approximated by whether the center of mass ever moved from between the feet or too far forward or backward. With this calculation as a heuristic, procedural footsteps could be taken in order to restore balance or the hips could be moved to maintain balance.

5 EVALUATION OF RESULTS

The results weren't amazingly accurate, but they weren't exactly expected to be. A lot of this was built on top of Unity's systems, and that caused some complications when figuring out how to do some things effectively. Specifically, built in IK passes made it difficult to modify the pose in some instances, so building separate IK system from scratch seems like it would allow for more flexibility.

6 FUTURE WORK

A significant part of this project was built on top of Unity's animation and IK systems, so the next step would be to try and create a new and separate system so that it is easier to define multiple IK passes as well as improve continuity between poses. Having more control over the IK passes would allow for more experimentation with constraints as well as allow for positional corrections to avoid intersections of the body with itself or the environment. There's still a lot more experimentation that can be done with constraints, both in how bio mechanical constraints in one area of the body can can effect the pose in another area, as well as in how soft constraints can be used and varied in order to achieve better posing.

Estimating and animating locomotion is another potential extension. Animations that had large lateral movements and significant weight shifts were not expected to work particularly well here, but extending the body pose solver to animate foot steps and attempt to estimate the pose based on velocity, acceleration, and inertia could create convincing motion even with a limited number of sensors

Finally, one of the ideas that I'm curious about is how machine learning may be able to be applied to the problem. Specifically, I

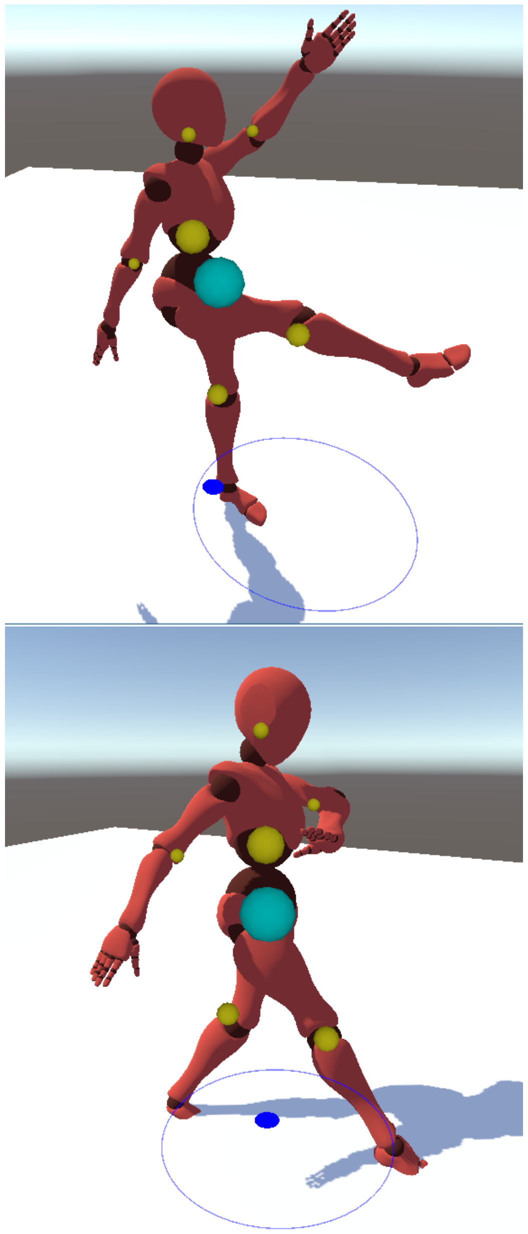

Fig. 4. These images show how center of mass (COM) and stability are calculated. The yellow spheres show the COM of each of the body segments, and the cyan sphere shows the COM of the entire body. The relative size of the spheres reflect their relative mass. The blue dot shows the COM's projection onto the ground, and the blue ring estimates where the weight of the COM can be without making the character appear unstable.

imagine that it may be possible to train a network to estimate the position of the elbow based on the position and rotation of the shoulder and wrist instead of using heuristics like joint relaxation.

As the accuracy of the body posing improves, doing user tests to see how the posing feels in first-person will be essential to understand how it could be applied. Currently, many VR experiences only use hand-models from the first-person view, so body pose tracking and animation could be used to create incrementally better first-person models that include the arms, shoulders, and possibly waist. Additionally, the full body pose could be useful in multi-player contexts as long as it can generate convincing poses, but that would require user testing to better understand what inaccuracies are tolerable.

7 CONCLUSION

The project didn't make any breakthroughs in terms of how body pose solving can be done in the future, but in trying to implement a solver I was able to better understand what kind of problems exist and how to overcome them in a future attempt. One of the most immediate things would be to write all of the IK passes from scratch to give better control over constraints and ordering since that seems to be essential after experimenting with heuristics.

ACKNOWLEDGMENTS

Thank you CSE 490V staff for all of the support throughout the course!

REFERENCES

Andreas Aristidou, Yiorgos Chrysanthou, and Joan Lasenby. 2016. Extending FABRIK with Model Constraints. *Comput. Animat. Virtual Worlds* 27, 1 (Jan. 2016), 35–57. https://doi.org/10.1002/cav.1630

Andreas Aristidou and Joan Lasenby. 2011. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graph. Models* 73, 5 (Sept. 2011), 243–260. https://doi.org/10.1016/j.gmod.2011.05.003

Sanghyun Kim, Junhyung Kim, Ji-Hun Bae, and Jaeheung Park. 2018. Real-time Inverse Kinematics Technique for Controlling Redundant Avatar Arm. https://www.researchgate.net/publication/329715304_Real-time_Inverse_Kinematics_Technique_for_Controlling_Redundant_Avatar_Arm

Pärtel Lang. 2016. Inverse Kinematics in Dead and Buried. http://root-motion.com/2016/06/inverse-kinematics-in-dead-and-buried/