Advanced Computer Graphics Spring 2019 (RPI)

Physics based Character Posing and Inverse Kinematics: Proposal

Frederick Choi

March 30, 2019

Introduction

Posing and animating rigs can be a pain for animators. Inverse kinematics solvers calculate the relative rotations of rigid bodies connected by different types of joints, which is useful especially for robotics and animation. However, when posing a rig for modeling or animation, some inverse kinematics solvers can produce unpredictable, and unintuitive, especially for large displacements. Inspired by how artists manually pose wooden armatures, I propose a physics-based character posing and inverse kinematics algorithm.

Cooper and Ballard [1] published a marker-based following algorithm that similarly constructs a physical model to simulate markers dragging the rig into position and use the Open Dynamic Physics engine to run their simulation. My algorithm will expand on this paper by focusing on posing. As described below, I will be implementing a simple physics engine and also be interpolating the motion of the control point to ensure predictable convergence.

iTasC published by Smith et al. [2] includes an algorithm for solving inverse kinematics, which is available in Blender. I will be comparing the performance of my algorithm with the iTasC algorithm in Blender based on the objective points below. Kulpa and Multon [3] have a hierarchical algorithm for IK on humanoid rigs that is both quick but also calculates realistic poses for humans. FABRIK by Artistidou and Lasenby [4] have an inverse-kinematics solver more geared toward robotics. These are both iterative methods that approximate the solution without transforming the problem into finding a minimum on a parameter space.

Pickl's master thesis [5] is a great resource for understanding rigid body dynamics. This paper will be the primary reference for rigid body dynamics.

Another advantage of the physics-based solver is that it is not an exact solver. That way, the target-points and constraints can be in impossible configurations, but the algorithm will find a pose that is a good compromise.

Objectives

The objectives of the physics-based character posing algorithm are:

- Accuracy. The end-effectors should end up close to their target position.
- Predictability. The solution should be close to the user's expectations.
- Intuitiveness. The interface should be easy to use

Accuracy can be evaluated quantitatively. Predictability and intuitiveness will be evaluated qualitatively by comparing it with other IK solvers in Blender. While speed is also a consideration, it will not be the focus of this project, since the limitations of the Blender plugin API makes it difficult to measure the true performance of the algorithm.

Algorithm

Interface. The posing tool will be implemented as a Blender plugin. The user will be able to place control points on the rig and move them around.

Control Points. A control point consists of two parts: the attachment point and the target point. The attachment point is on the rig, while the target point is what the user moves around. The attachment point is brought toward the target point by simulating a spring (linear, constant, or non-linear). Multiple control points will be supported.

Movement Interpolation. When the target point of a control point is moved far away from a control point (i.e. the angular displacement exceeds some threshold) then the movement of the target point will be interpolated over time to ensure the algorithm converges to a predictable solution. Some ideas to experiment with include:

- A circular arc around the center of mass
- A circular arc around the joint
- Linear time interpolation
- Ease-in / ease-out time interpolation

Physics. Physics will be implemented as well, to ensure things work with Blender, and to have finer control over the specifics. After all, the goal is to get a predictable final position. The accuracy of the physics in between is only important insofar as it affects the outcome. The mechanics I plan to model are:

- External torques
- Inter-joint forces
- Torsion forces

Project Timeline

The following is an outline of what I will accomplish each week.

Week 0. I already worked through the Blender API and I have a simple plugin working. I also currently have things moving, and multi-joint physics working on the plane.

Week 1. Get multi-joint physics working in 3D space and get axial torques to work. Model rotation constraints as counter-rotation springs. To demonstrate, I will pose a simple two-joint model (in real time).

Week 2. Get physics working on branching/multi joints (hip/shoulders). To demonstrate, I will pose a simple field-goal shaped model.

Week 3. Work on interpolation of marker movement and play around with different types of springs. To demonstrate, I will have comparisons between several of the above alternatives.

Presentation. Live demo of posing a simple human skeleton (limbs, torso, no fingers).

References

[1] J. L. Cooper and D. Ballard, "Realtime, physics-based marker following," in *Motion in Games* (M. Kallmann and K. Bekris, eds.), (Berlin, Heidelberg), pp. 350–361, Springer Berlin Heidelberg, 2012.

[2] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, *iTASC: A Tool for Multi-Sensor Integration in Robot Manipulation*, vol. 35, pp. 235–254. 03 2009.

[3] R. Kulpa and F. Multon, "Fast inverse kinematics and kinetics solver for human-like figures," in *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pp. 38–43, Dec 2005.

[4] A. Aristidou and J. Lasenby, "Fabrik: A fast, iterative solver for the inverse kinematics problem," *Graphical Models*, vol. 73, pp. 243–260, 2011.

[5] K. Pickl, "Rigid body dynamics: Links and joints," Master's thesis, University of Erlangen-Nuremberg, 9 2009.