CS6665 - Character Animation
Project 1 - Inverse Kinematics

Duong Hoang

October 6, 2014

1 Introduction

In this project, I implement an inverse kinematics (IK) solver using the damped least square (DLS) method. Compared to other popular methods for solving IK problems such as Jacobian transpose or pseudo-inverse, DLS is more stable near singularities, which means the animation is less jerky when the arms are fully stretched. The solver supports any combination of translational joints (1-dof) and rotational joints (1/2/3-dof). It can track multiple targets at the same time, while maintaining realtime framerates (60fps) for moderately complex bodies. Simple, per-dof joint limits are supported. The solver can also maintain a constraint on the horizontal position of the system's center of mass.

2 Inverse Kinematics as a Linear Problem

Given a body which consists of a a hierarchy of joints, we want to rotate (or translate) the joints so that certain pre-defined locations on the body (called end effectors) closely track some (possibly moving) targets in the outside world. If we use a column vector $\boldsymbol{\theta}$ to denote all the joint angles ($\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_n]^T$, where n is the number of degrees of freedom of the whole body), $\boldsymbol{s}$ for the x-y-z coordinates of all end effectors ($\boldsymbol{s} = [s_1, s_2, \ldots, s_m]^T$, where $m/3 = k$ is the number of end effectors), $\boldsymbol{t}$ for the x-y-z coordinates of all targets ($\boldsymbol{t} = [t_1, t_2, \ldots, t_m]^T$), then $\boldsymbol{s}$ is a function of $\boldsymbol{\theta}$, and the goal of IK is to find $\boldsymbol{\theta}_i$ so that $\|\boldsymbol{s}(\boldsymbol{\theta}_i) - \boldsymbol{t}\|$ is minimized.

To relate the change in $\boldsymbol{\theta}$ with the change in $\boldsymbol{s}$, we use a Jacobian matrix $\boldsymbol{J}$, defined as $\boldsymbol{J}_{ij}(\boldsymbol{\theta}) = \partial \boldsymbol{s}_i / \partial \boldsymbol{\theta}_j$. Then we can approximate the change in $\boldsymbol{s}$ as $\Delta \boldsymbol{s} \approx \boldsymbol{J} \Delta \boldsymbol{\theta}$. Using $\boldsymbol{e}$ to denote the difference between the targets and the end effectors ($\boldsymbol{e} = \boldsymbol{t} - \boldsymbol{s}$), we can solve the IK problem by finding an update $\Delta \boldsymbol{\theta}$ to add to the current $\boldsymbol{\theta}$ so that the $\boldsymbol{e} \approx \Delta \boldsymbol{s} \approx \boldsymbol{J} \Delta \boldsymbol{\theta}$, that is, solving for $\Delta \boldsymbol{\theta}$ in the equation

$$\boldsymbol{e} = \boldsymbol{J} \Delta \boldsymbol{\theta}. \tag{1}$$

For details of computing $\boldsymbol{J}$, please refer to [Bus04].

3 Damped Least Square Solver

Very often, Equation (1) does not have an exact solution. It is common to solve for $\Delta\boldsymbol{\theta}$ such that $\|\boldsymbol{J}\Delta\boldsymbol{\theta} - \boldsymbol{e}\|$ is minimized instead, that is, solving the normal equation

$$\boldsymbol{J}^T\boldsymbol{J}\Delta\boldsymbol{\theta} = \boldsymbol{J}^T\boldsymbol{e} \tag{2}$$

It can be shown that Equation (2) always has a solution. However, the solution is unstable near singularities, which can cause jerkiness in the animation. The damped least square method addresses this problem by solving a similar equation

$$(\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})\Delta\boldsymbol{\theta} = \boldsymbol{J}^T\boldsymbol{e} \tag{3}$$

It can be shown (by SVD decomposition) that $(\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})$ is always invertible, hence Equation (3) always has a solution

$$\Delta\boldsymbol{\theta} = (\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{4}$$

The same SVD decomposition can also show how Equations (3) and (2) are related, and why using (3) leads to a more stable solution [Bus04]. Instead of solving 4, we note that $(\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{J}^T = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I})^{-1}$ and use the following formula to solve for $\Delta\boldsymbol{\theta}$

$$\Delta\boldsymbol{\theta} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{e} \tag{5}$$

The advantage of using (5) over (4) is that the matrix $\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I}$ has size $m \times m$ instead of $n \times n$ (recall that $m = 3k$ where k is the number of end effectors/targets, and thus usually is much less than n, the total number of degrees of freedom in the body). We use row operations to find $\boldsymbol{f}$ such that $(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I})\boldsymbol{f} = \boldsymbol{e}$, and then $\Delta\boldsymbol{\theta} = \boldsymbol{J}^T\boldsymbol{f}$ is the solution.

4 Joint Limits

For each degree-of-freedom i we define a limit range: from θ_i^{min} to θ_i^{max}. If a solution falls out of this range, it is simply clamped. This way of handling joint limit is quick and reasonable for 1-dof and 2-dof joints. For 3-dof rotational joints (ball joints), it probably makes more sense to define the limit as a cone of possible orientations. Another problem is that our system will sometimes miss legitimate solutions by not considering the joint limits when solving for $\Delta\boldsymbol{\theta}$. This problem, however, is not too severe if the targets' trajectories are smooth (i.e. targets do not jump), as they often are in practice.

5 Constraining the Center of Mass

Assuming the body has a uniform mass distribution, we can solve the IK problem while trying to keep the projected center of mass (cof) of the body at one pre-defined point on the horizontal plane. The idea is to formulate the position of the cof ($\boldsymbol{c}$) as a linear function of the joint angles ($\boldsymbol{\theta}$), then use a Lagrange-multiplier-based method to solve for an optimal $\Delta\boldsymbol{\theta}$ that also satisfies the constraint, similar to what is done in [Gle98].

Denote the Jacobian relating the change in $\boldsymbol{\theta}$ with the change in $\boldsymbol{c}$ as $\boldsymbol{J}_c$. The constraint can be written as

$$\boldsymbol{J}_c \Delta\theta = \Delta\boldsymbol{c} \tag{6}$$

Taking the derivative of Equation (2), and set it to a linear combination of the derivative of the constraint (6), we have

$$\Delta\boldsymbol{\theta}^T(\boldsymbol{J}^T\boldsymbol{J}) - \boldsymbol{e}^T\boldsymbol{J} = \boldsymbol{\lambda}^T\boldsymbol{J}_c \tag{7}$$

where $\boldsymbol{\lambda}$ is a column vector of Lagrange multipliers. Taking the transpose of (7), and adding a small term $\epsilon\boldsymbol{I}$ to $\boldsymbol{J}^T\boldsymbol{J}$ to make the matrix invertible, we arrive at

$$(\boldsymbol{J}^T\boldsymbol{J} + \epsilon\boldsymbol{I})\Delta\boldsymbol{\theta} - \boldsymbol{J}^T\boldsymbol{e} = \boldsymbol{J}_c^T\boldsymbol{\lambda} \tag{8}$$

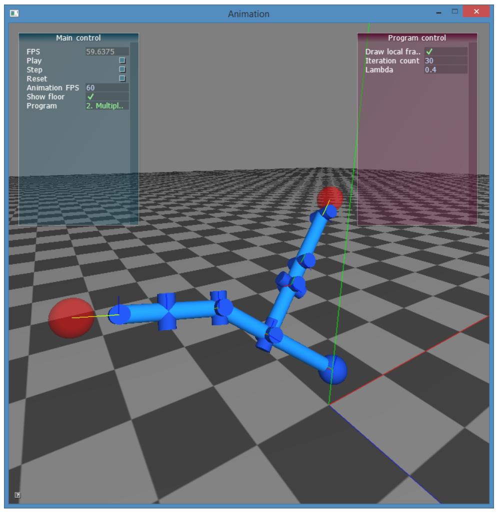
Take out $\Delta\boldsymbol{\theta}$ in (8) and substitute it into (6) to get

$$\boldsymbol{J}_c(\boldsymbol{J}^T\boldsymbol{J} + \epsilon\boldsymbol{I})^{-1}(\boldsymbol{J}_c^T\boldsymbol{\lambda} + \boldsymbol{J}^T\boldsymbol{e}) = \Delta\boldsymbol{c} \tag{9}$$

Equation (9) is a linear equation in the form of $\boldsymbol{A\lambda} = \boldsymbol{b}$, which can be solved using the damped least square method described in Section 3. Once $\boldsymbol{\lambda}$ is found, it can be substituted back into (8) to solve for $\Delta\boldsymbol{\theta}$.

6 Experiments and Discussion

Here we describe the implementation of our system, and a few notable experiments we have done. Our IK program lets the user control the animation by pausing and stepping through the time steps, or speeding up/slowing down the animation. In our implementation, $\Delta\boldsymbol{\theta}$ is solved for a number of iterations per animation frame. The user can adjust this number as well as the damping factor λ used in DLS.

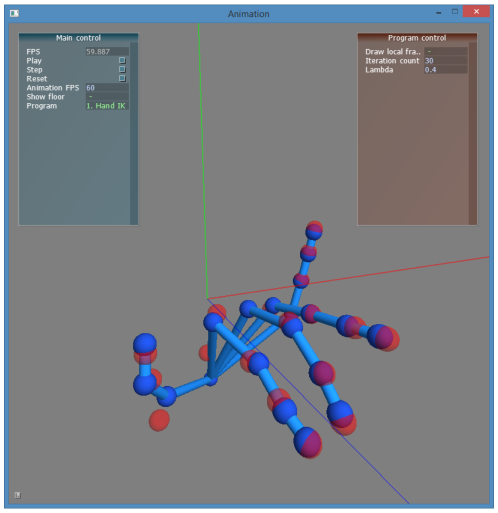


FPS 59.6375
Animation FPS 60
Show floor ✓
Program 2. Multipl..

Draw local fra.. ✓
Iteration count 30
Lambda 0.4

Figure 1: Multiple joint types

Figure 1 shows the interface of our program, and an example of a body consisting of one ball joint (root), three universal joints, and two hinge joints, tracking two moving targets. The targets often move out of reach of the body, but the animation is smooth, given a high enough damping factor.

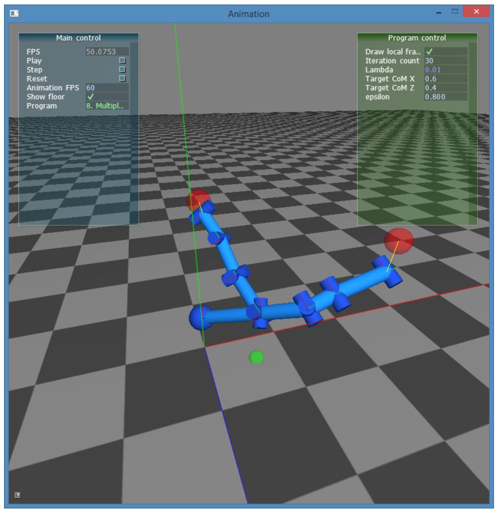


FPS 59.887
Play
Step
Reset
Animation FPS 60
Show floor -
Program 1. Hand IK

Draw local fra.. -
Iteration count 30
Lambda 0.4

Figure 2: Hand skeleton driven my mocap data

Figure 2 shows a skeleton of a hand, with 20 ball joints and a movable root, tracking 20 targets, specified exactly at the positions of the joints. The targets' trajectories and the hand's skeleton are taken from motion capture data. Our system gives stable solutions and maintains real-time framerates for this case.



FPS 50.0753
Play
Step
Reset
Animation FPS 60
Show floor ✓
Program 8. Multipl..

Draw local fra.. ✓
Iteration count 30
Lambda 0.01
Target CoM X 0.6
Target CoM Z 0.4
epsilon 0.800

Figure 3: Center of mass constraint

Figure 3 shows the same body as in Figure 1, but with an added center-of-mass constraint (shown as the green sphere on the floor). The system is able to track the moving targets while maintaining a center of mass over the specified position. The user has control over the damping parameters used to solve the contrained optimization (ϵ and λ). We have noticed that in general ϵ (see Section 5, Equation (8)) controls the stability of the solution, while λ (the DLS damping factor) controls how well the constraint is maintained.

In conclusion, we have found that DLS is a stable and efficient method for solving the IK problem. The drawback, however, is that the damping factor λ must be chosen by hand. We could obtain better results if different λ values are used for different joints, but this implies the need for a reliable way of choosing these damping values. This approach is explored in [BK04].

References

[BK04] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10:37–49, 2004.

[Bus04] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, IEEE Journal of Robotics and Automation, 2004.

[Gle98] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 33–42, New York, NY, USA, 1998. ACM.