**CM50244: Computer Animation and Games 1**

**Yongliang Yang**

**Coursework 1: Inverse Kinematics**

**Lander King: MComp Computer Science and Mathematics**

1. **2D Linkage and Forward Kinematics**

To be able to explore inverse Kinematics using a linkage in 2D space I first needed a suitable representation of a bone/rod. The Basics of this requires a bone to have a Starting Position, Length and an angle representing the rotation of a basic hinge at the base of the bone, with this information we can then work out the end location of the bone using simple trigonometry





With this information we can then combine the bones together to form a linkage. Where each bone has a child, whose starting position is the same as the end location of the parent bone. with all this work done it would now be possible to implement forwards and inverse Kinematics

To run forward kinematics on the linkage, we set the angle of each bone in the linkage to the desired angle and re-compute its end location. Starting at the root of the linkage and recursively moving down to the linkage until you reach the final bone.

For this to work recursively we store two angles in each bone the selfAngle and the jointAngle. selfAngle represent the angle difference from the itself to its Parents Angle, this is the angle we increment whenever we want to animate the process. alternatively, jointAngle represent the angle the bone is from the X-axis and is used when calculating the end position of the bone, it is calculated by adding together the selfAngle and the parents jointAngle.



1. **Inverse Kinematics**

The goal of Inverse Kinematics(IK) is to find the angles of the linkage when the end position of the final bone called the end effector, is at a desired point.

The method I have used to solve this problem is an iterative gradient decent algorithm which tries to minimise the distance from the end effector to the desired location in each iteration. It does this by calculating the Jacobian matrix of the linkage which can be defined as the following for a linkage with 3 bones

where a is the axis the system is rotation about, in a 2D system this will be (0,0,1), E is the location of the end effector and is the position of the i-th bone. The Jacobian represent how much of the end effectors position is affected by the rotation of each bone.

With the Jacobian matrix, we can now calculate the difference in angles dθ which will hopefully move the linkage closer to the goal.

where V is the change in spatial location need for the end effector to be at the desired location. As the inverse Jacobian may not exist, we can approximate it be using either the pseudoinverse of the Jacobian or the Jacobian transpose. I will be compare the usage of both to test how much of an effect it will have on the performance of the IK.

With we can calculate the new anglesof the linkage

The value s is used to scale the step in the direction of the desired angle, this help the algorithm from over stepping and moving further away from the desired location. In my system I have set s = 0.025 which solves most problems in a good number of iteration while still achieving a good result. This variable could be changed if we wanted different results. To stop overstepping from happing, whenever the linkage moves further away from the goal in an iteration (overstep), I will then half the current step size.

To calculate whether to stop the iterative process I, calculate the mean square error (MSE) between the desired location and the end effector position, once this error function is minimised to a reasonable amount the iterative process will stop. In my system this has been set that the MSE must be lower than 0.001 for the process to terminate. Another stop mechanism I use, is to check if the Euclidian distance moved each iteration is over a suitable amount, if it hasn’t then it means the linkage has stopped moving towards the desired location. This will only occur when the desired location is a location the linkage cannot reach, I have set this value to 0.0001.

* 1. **Choice of Jacobian and Step Size**

To test the effect of the choice of the substitute of the Jacobian Inverse in the calculation of , I ran some test on my IK system, measuring the number of iterations required to find the solution and the final error at the end on multiple IK runs with different goal locations.

I tested this on a 4-bone system with the root bone having length 10 and each child having the length 0.8 \* L, where L is the Length of its parent.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal | Pseudo Inverse | | Jacobian Transpose | |
| Iterations | MSE | Iterations | MSE |
| [-10,-20] | 230 | 0.002 | 110 | 0.00094 |
| [0,-20] | 223 | 0.002 | 30 | 0.00082 |
| [10,-20] | 527 | 0.0039 | 68 | 0.00096 |
| [-10,0] | 241 | 0.0019 | 16 | 0.00041 |
| [-10,20] | 267 | 0.0019 | 64 | 0.00089 |

Even with this limited test it is fair to say that using the Jacobian Transpose is a far more efficient and accurate method, also it is computationally more efficient as calculating the Transpose often easier than calculating the pseudo inverse.

I also want to test the effect of the choice of step size used, to do so I will run the same IK tests above but with altering step size values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal | Step Size | | | |
| 0.1 | 0.05 | 0.025 | 0.01 |
| Iterations | Iterations | Iterations | Iterations |
| [0,-20] | 98 | 78 | 110 | 53 |
| [-10,-20] | 62 | 74 | 30 | 50 |
| [0,-20] | 94 | 74 | 68 | 52 |
| [10,-20] | 47 | 11 | 16 | 29 |
| [-10,0] | 80 | 73 | 64 | 91 |

As you can tell from the table, the number of Iterations decrease when you decrease the step size in the IK. Once the step size if very small eg 0.01 you get to the point that the number of Iterations is going to increase.

Another interesting feature of altering the step length, I the final shape of the linkage is different for different values of the step size, this is a nice observation that the solution to a give IK problem is not a unique solution and can have many possible angle compositions. As demonstrated below

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA close up of a map

Description generated with very high confidence

*Each picture represents the IK solution for G = [0,-20] with different step lengths.*

*s =0.1 (top left)*

*s = 0.01 (top right)*

*s= 0.001 (bottom left)*

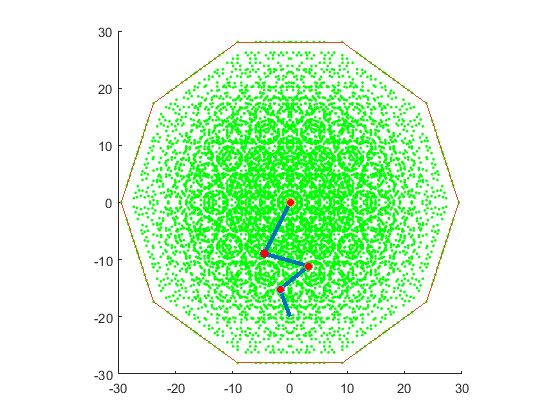
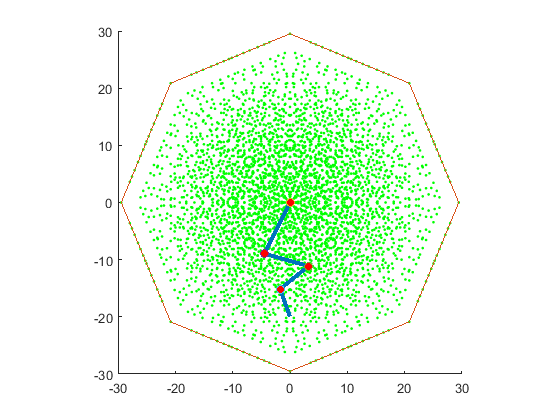
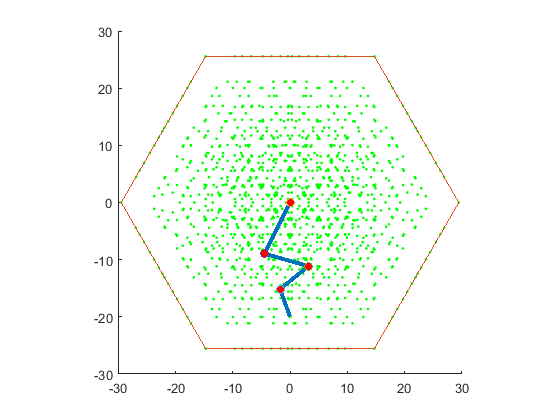
1. **Inverse Kinematics Solution Space**

With IK it is possible to not know the area in which the end effector can reach especially with very complicated skeletons eg. A skeleton system representing the bone structure of a 3D character or with a linkage that has joint constraints. Hence it would be very useful to be able to somehow visualize this area.

To do this for my linkage system I find the solution space of the end effector by Iterating through all valid combinations of angles for the linkage. As the set of values that the angle can be assign to in each bone is a uncountable set, we need to create linearly space vector that we can iterate through and save the location of the end effector. The spacing of the linspace can be defined by the user.

As a massive group of points drawn on a screen is not very appealing or useful, I then find the boundary that contains of the found points. This gives a nice representation of the solution space.

This method of visualising the solution space is costly as it has O() complexity where n is and k is the number of bones in the linkage. So is not feasible for large linkages, but for this assignment it works. Also, this method fails to find the inner boundary of a linkage, which sometimes occurs if the Child bones of the root bone cannot be arrange such that the end and start have the same position.



1. **Physical Properties**
   1. **Rate**

To be able to change the rate of rotation of each bone, all that needs to be bone is a let the user set the InterpRate variable of the bone. This variable then changes the rate at which the angle is updated when animating the linkage.

* 1. **Slow in - slow out**

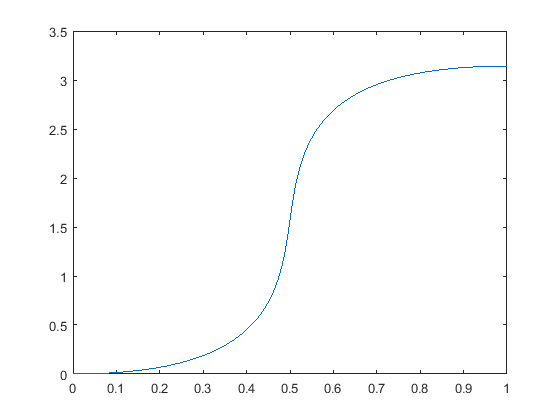
To implement slow-in, slow-out animation, first need to switch from linear interpolation of the angle to Bezier interpolation, and then let the user define the positions of the control points such that it will give a slow-in, slow out effect. This happens when the control points positions are of the form



Where ControlPoint is a variable defined by the user stating what value of time the first control point of the Bezier curve is going to be defined.

Adding variable rate to the Bezier curve interpolation involves finding the number of steps N required to linearly interpolate to the new angle, then calculating the values of the Bezier Curve in the same number of intervals. Then to animate along the Bezier curve I calculate the gradient of the approximated curve, then move the angle of the bone by that much.



****

*Bezier Curve of the angle moving from 0 to pi at an Interpolation rate of 0.05. The Control Points are defined at (0.9, 0) and (0.1, pi)*

* 1. **Min and Max Angles**

To add joint constraints to the bones, involves first adding variables that represent the maximum clockwise and anticlockwise rotations. Then why animating the linkage you need to check if the desired angle is within the range defined. This involves comparing the clockwise and anticlockwise rotations needed to reach the desired angle and the rotations needed to get the maximum and minimum angles. To make sure angles stay in a useful range I bound them from 0 to 2 PI

If the angle is within this range, take the shortest path such that you don’t go past the max or min angles. If you can’t reach this angle, then move to the max or min angle that is closest to the desired angle.



With Linkages with joint constraints, you need to adapt how the IK handles certain situations. To remedy this, we make sure the first thing you do to the linkage is move it such that all joints are at the mid value. This means that once you have calculated the linkage will only move in the direction that minimises the distance from the end effector and desired location within the range of the angles. Then if it wants to move past the max or min angle you then fix the bone in place.

1. **Conclusion**

In this assignment I have explored many aspects of building a basic linkage system capable of computing forward and inverse Kinematics in real time. Even though I have added many features, it is still a basic system and has many limiting factions. Such limiting factors involve not being able to find the complete solution space given a linkage system, in a real-time scenario.

My system also is currently limited to only having 1 child bone, which isn’t often the case of most skeletal systems. By adding support of multiple children bones you could start representing much more complicated bone structures, which in highly useful in computer animation and games.

Another main area by system does not incorporate is the modelling and varying the flexibility of the bones in the system. With flexible bones you could slightly bend/ stretch bones to slightly more interesting shapes compared to rigid bones. But in doing so would increase the computation of the Inverse Kinematic algorithm as it would now have to calculate a more complicated Jacobian matrix with more variables to find. Modelling the flexibility of bones in my eyes is not that important as the technique to use, as most real bones are not that flexible, so its use would be limited. Also, it would be possible approximate flexible bones by splitting a bone into more bones that are smaller, that are constrained to a limited angle range.

In the assignment I have only explored one method of Inverse Kinematics, but there are more sophisticated methods available that solve the problem quick and easier such as Cyclic Coordinate Decent (CCD) that I didn’t get to explore

If I was to continue this assignment I would like the implement a method of animating the linkage from one location to the other such that all the bones rotate and the same time. This would produce a more natural looking animation than I currently get.

Even though there are many limitation and improvements that can be made, and more topics that could be explored. Overall, I’m pleased with the state of IK and linkage system and the work I have complete on this assignment