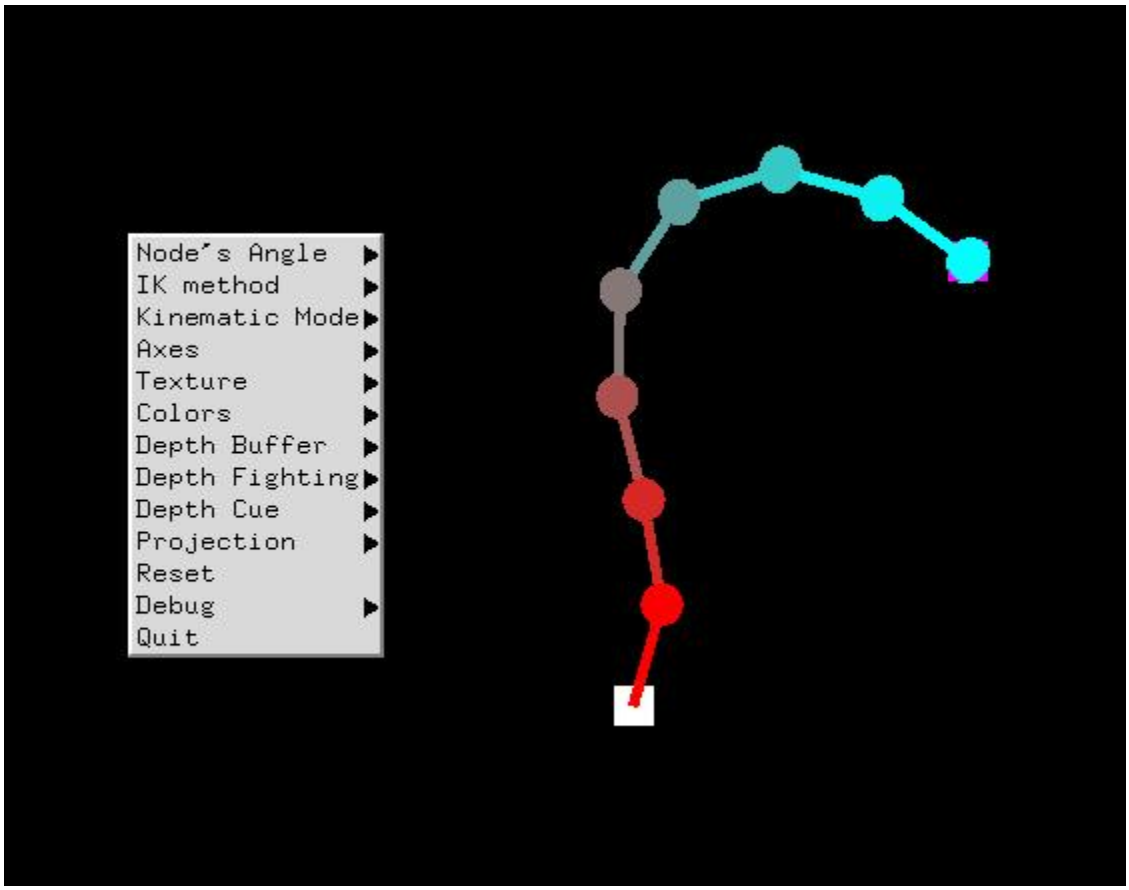
CS419 Inverse Kinematic Report

Tianle Yuan, Shiyao Wang

yuant@oregonstate.edu, wangshiy@oregonstate.edu

Part I.The Forward Kinematics

For the program, a basic interface was created to handle the user input. The instruction was as following:



When right click on the program, a menu will pop out. For this project, only the first three menu will be important:

Node's Angle: You can choose to change the angle of a specific node, when chose, you will be prompt a input for the new angle on the terminal.

IK Method: Choose to use CCD or SVD method for Inverse Kinematic.

Kinematic Mode: Choose to use Forward or Inverse Kinematic.

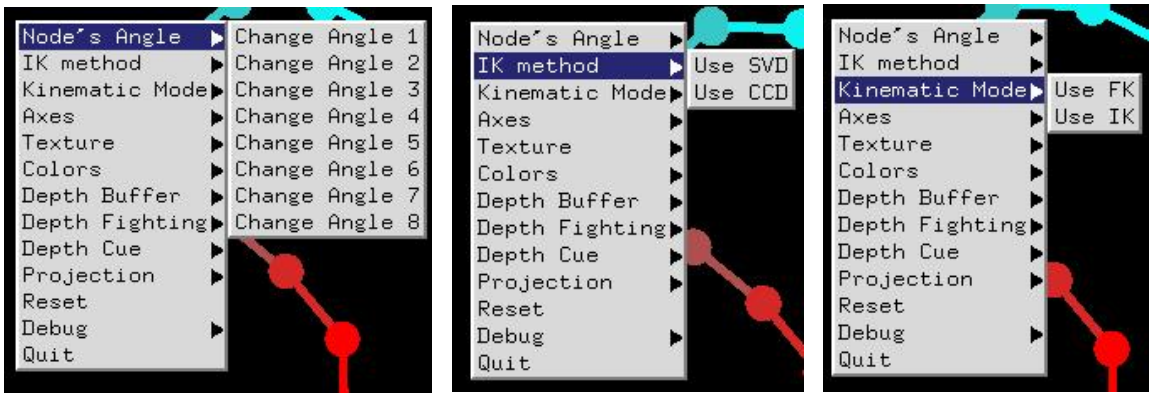


Fig 2. The example of three menus

As for part 2 of the assignment, you need to choose the FK mode and then you can start to change value for specific node:

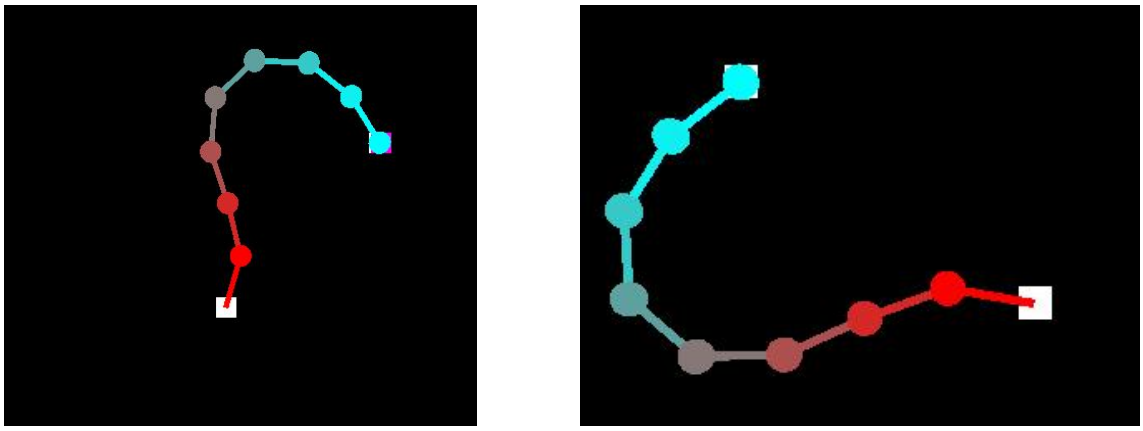


Fig 3. The angle of node 1 was set to be 170

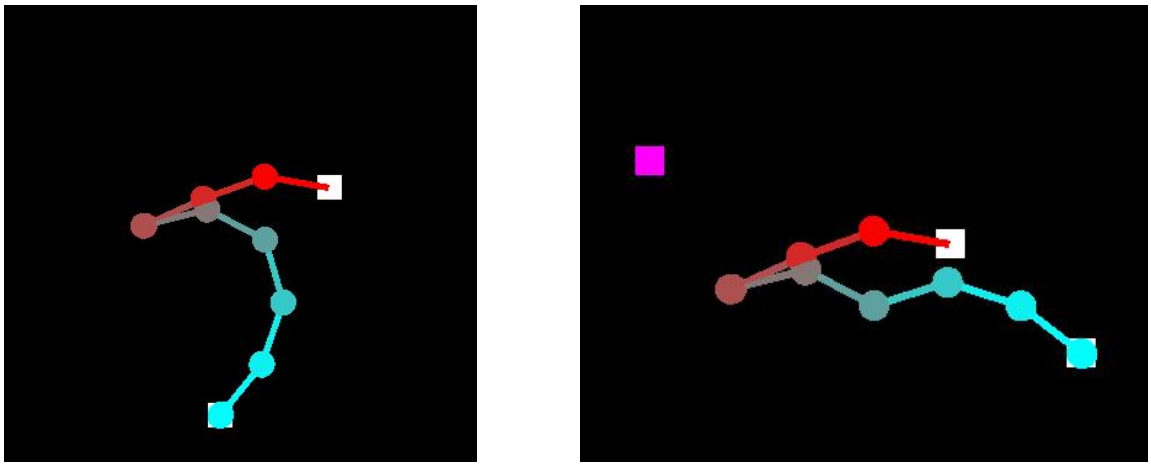


Fig 4. The angle of node 4 was set to 170, and then the angle of node 6 was set to 45

The all 4 screenshot was continuous, thus the entire rooter node's change will affect all the rest.

Part II. The Inverse Kinematics

For this part, you will need to choose the IK mode and you have two different options to use.

1. The CCD Method

   The main idea of CCD Method is to iterate through the whole list, and for each single arm it will calculate a rotation that make the current root-to-tip vector to match the root-to-target vector. One thing need to mention here is that this "root" is not the whole arm list's root, it's the current arm's root in the world coordinates system.

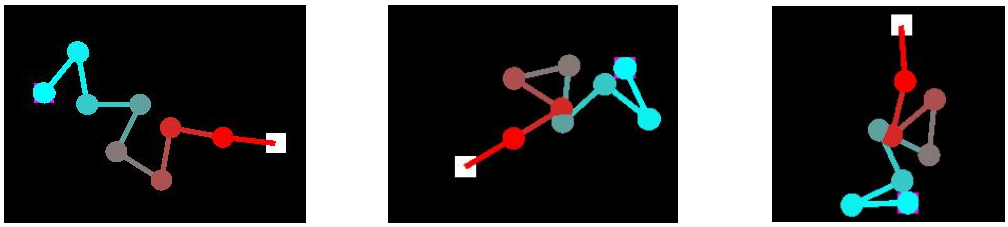


Fig 5. *The CCD method for reachable target point (purple square, white square is the fixed origin)*

The method is comparatively "correct-er" than the SVD method, thus when coming to the singular system problem, CCD method will give you a guaranteed method that move the tip as close as possible to the target.

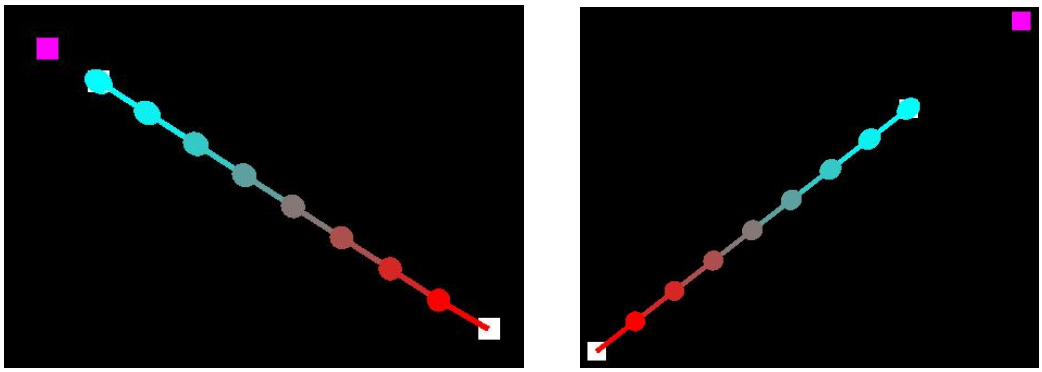


Fig 6. *The CCD method for unreachable point*

When finding a solution for unreachable point, the CCD method will eventually made the whole arm system into a static position, which is quite different with the SVD method.

But when coming to the situation that **number of arms * arm length >>>> target distance**, the CCD method tending to generate a special "circle flower" pattern, which I personally think was result by its iterate-rotation solution of the IK problem:

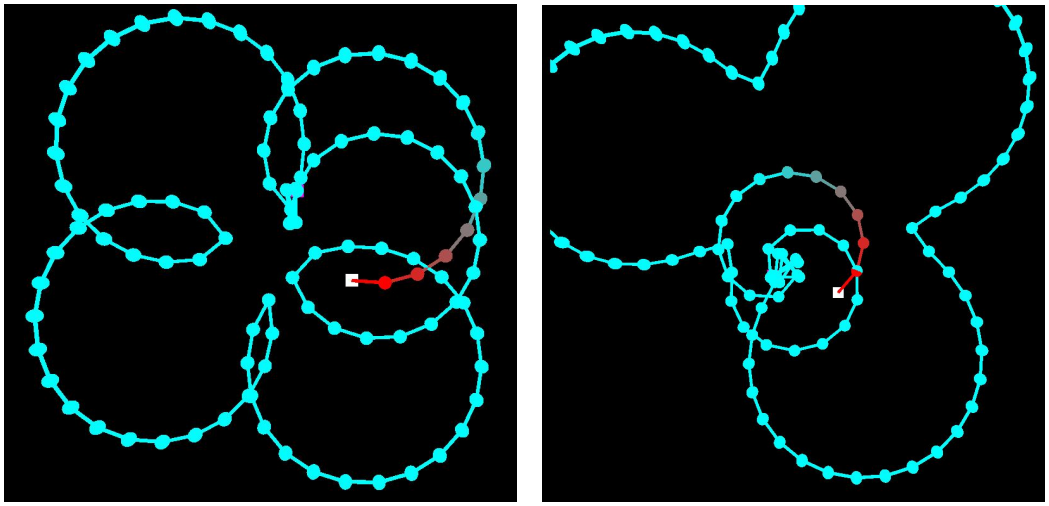


Fig 7. *A "Flower Pattern" when set number of arms to be 100.*

2. The SVD Method

The second method is a more numerical method. By forming a Jacobian Matrix, the solution of the IK problem could be found by applying the matrix **J**'s pseudoinverse matrix with the target orientation of the whole system, which could be solved by SVD method. For this project, I used the SVD method of the Eigen library specifically.

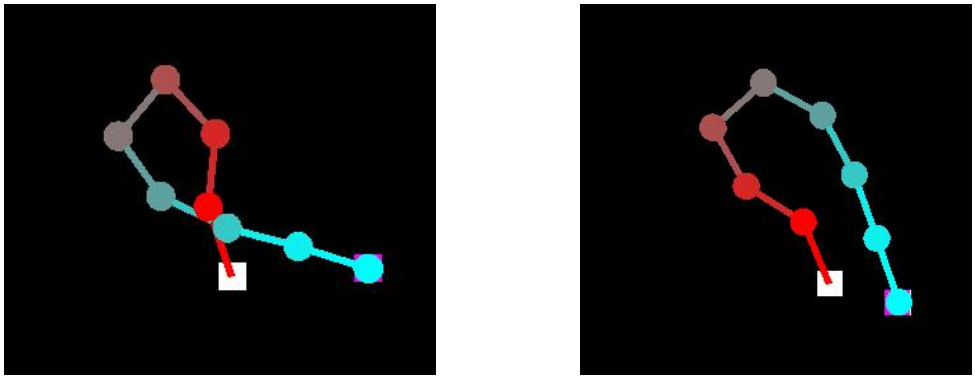


Fig 8a. *As the last target point was at left of the origin, the SVD method tending to give a solution that "twisted" the whole arm system (because not all the arm in SVD method has to move a lot), as the CCD method always move all the arms (since it actually iterate all the arms).*

Fig 8b. *When the target point is reachable, the SVD tending to generate the method that "bending" the whole arm system, rather than "crazy-twisty" result when using CCD method (as showed in Fig. 5).*

But when calculating the solution for unreachable target point, the SVD method will keep trying to find the solution to the point, which result in a super-quick-tremble action.

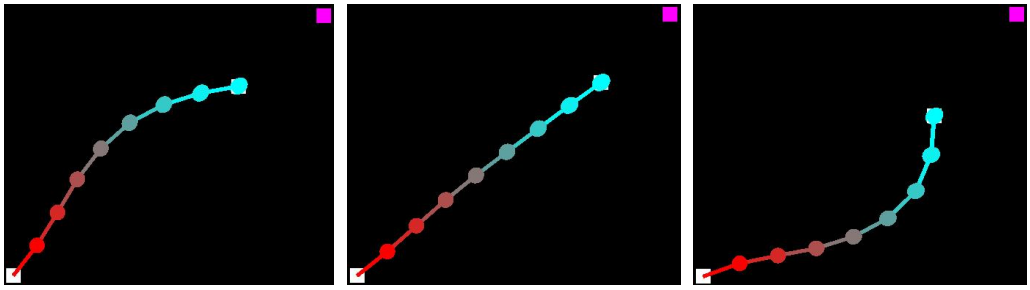


Fig 9. *The trembling action when trying to reach a unreachable target point, while the CCD method will eventually result in a static position.*

Update: After I modified the algorithm to cast the unreachable target point within the reachable distance, the SVD algorithm can now also handle the unreachable point with a stable solution without "shaking".

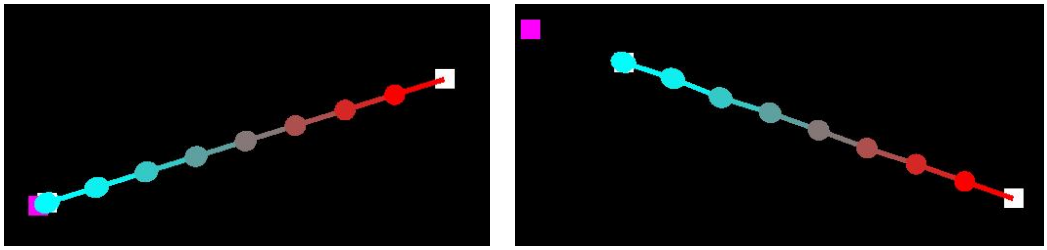


Fig 10. *The fixed SVD algorithm with a stable solution for unreachable point, just as the CCD solution.*

One big difference of SVD with CCD is that when in the situation that **number of arms * arm length >>>> target distance**, it will tends to generate a "spring pattern":

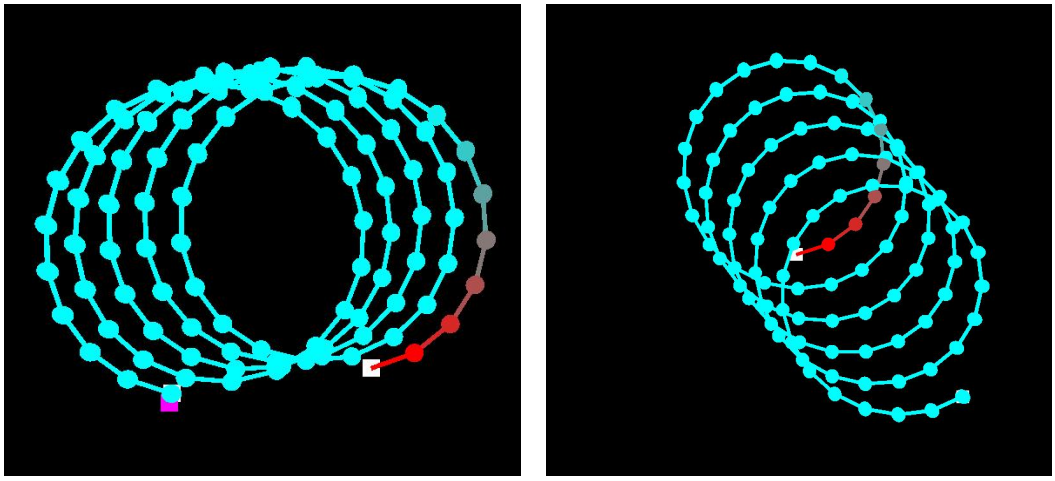


Fig 11. *A "Spring Pattern" when set number of arms to be 100.*