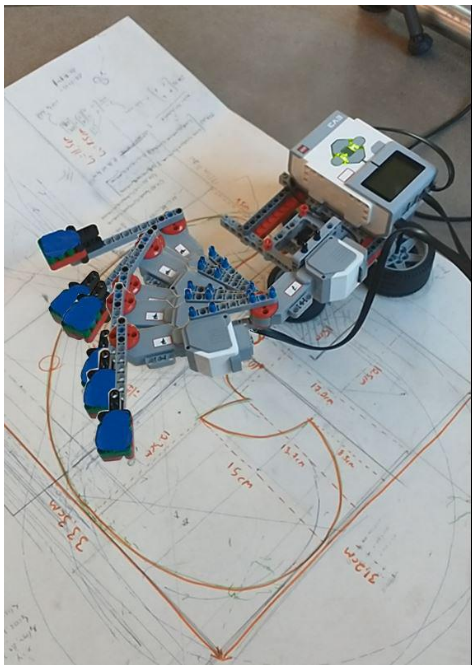
Lab 3
October 27 2021
Joshua Jarman
Nathan Pelletier

1. Plath Planning:

a) Draw a line based on two points

The math used in path planning was simple, $y = mx + b$, $m = y_1-y_2/x_1-x_2$
We chose our step size based on a constant value applied to our x which we then calculated y with. As shown by the picture below our program moved the arm in a semi straight line.



Included in our folder are 3 videos displaying the general accuracy of our machine for 5 steps, 10 steps and 20 steps respectively. We found that smaller step sizes resulted in more jagged lines with the robot doubling back over old points on its path. This could be a bug in our programming but I think it is more likely that the angles the robot must drive are too small to effectively move.

b) Draw a line based on a point, an angle and a distance

We reused the base code above so our results were the same. Our formulas changed slightly as we converted the distance into its x and y components before adding them to our base point.

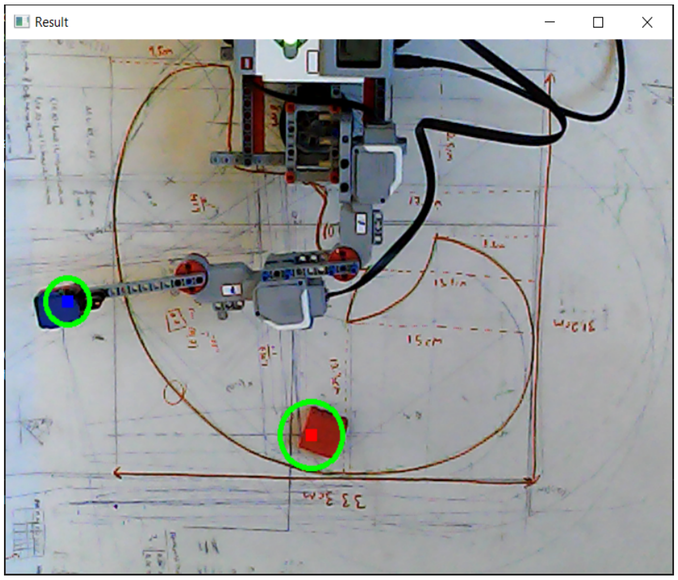
c) Draw an arc based on n points

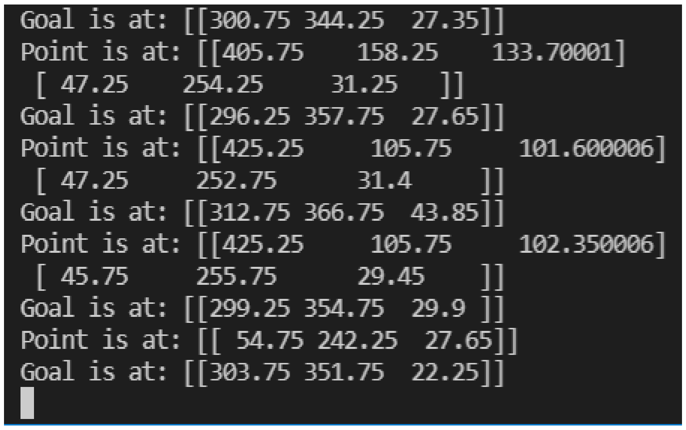
While our drawArc function works for n points, I (Nathan) feel we failed to understand the question. If there were three points we could calculate the arc based on point 1 and point 3 being the length and point 2 being the general arc radius.

Lab 3
October 27 2021
Joshua Jarman
Nathan Pelletier

2. Uncalibrated Visual Servoing

This part of the assignment stumped us. We managed to receive visual information and while the server booted correctly we were unable to get the client to work. Below are images of our visual servoing receiving visual information.





```
Goal is at: [[300.75 344.25  27.35]]
Point is at: [[405.75    158.25    133.70001]
 [ 47.25    254.25     31.25   ]]
Goal is at: [[296.25 357.75  27.65]]
Point is at: [[425.25     105.75     101.600006]
 [ 47.25     252.75      31.4     ]]
Goal is at: [[312.75 366.75  43.85]]
Point is at: [[425.25     105.75     102.350006]
 [ 45.75     255.75      29.45     ]]
Goal is at: [[299.25 354.75  29.9 ]]
Point is at: [[ 54.75 242.25  27.65]]
Goal is at: [[303.75 351.75  22.25]]
```

On the client side we simply started the client port and ran a while loop waiting for button press to kill the program. It should have read inputs from the server and moved the motors to those angles.

Our server side code was where we did the heavy math. We wiggled our arm a set amount and used the resulting ratios of position/angle to calculate our first Jacobian. We then found an average goal so that our goal became more stationary with each loop. We updated our Jacobian using the broyden update.
The picture below shows Nathan putting random values in to test the effectiveness of the said update.

Lab 3
October 27 2021
Joshua Jarman
Nathan Pelletier

$$\begin{bmatrix}4&3\\2&1\end{bmatrix} + \frac{\left(\begin{bmatrix}15\\3\end{bmatrix} - \begin{bmatrix}4&3\\2&1\end{bmatrix}\begin{bmatrix}5\\5\end{bmatrix}\right)\begin{bmatrix}5&5\end{bmatrix}}{\begin{bmatrix}5&5\end{bmatrix}\begin{bmatrix}5\\5\end{bmatrix}}$$

$$\begin{bmatrix}15\\3\end{bmatrix} - \begin{bmatrix}35\\15\end{bmatrix}$$

$$\frac{\begin{bmatrix}-20\\-12\end{bmatrix}\cdot\begin{bmatrix}5&5\end{bmatrix}}{50}$$

$$\frac{\begin{bmatrix}-100&-100\\-60&-60\end{bmatrix}}{50}$$

$$\begin{bmatrix}4&3\\2&1\end{bmatrix} + \begin{bmatrix}-2&-2\\-12&-1.2\end{bmatrix} = \begin{bmatrix}2&1\\0.8&-0.2\end{bmatrix}$$

Given [2, 1] * [5, 5] = 15, [0.8, -0.2] * [5,5] = 3.
It seems the Broyden update morphs our Jacobian to correctly perform the last calculation only, shifting values slightly each time to better approximate the larger function (5x + 5y = 15, x can be any value with a fixed y).
With our Jacobian we found our incorrect but close angles and in theory moved our arm. Repeating the update of our Jacobian and trying again.We decided that our stop condition should be plus or minus 25 pixels in the x and y direction as our visual feedback could no longer identify our arm from the goal below that range.

3. Explain what modifications would need to be made to your UVS program, equipment setup, and mathematical equations in order to control and move a robot arm in a 3D coordinate system (x, y, z location).

Program: As our Jacobian would change drastically to accommodate the new dimension we would have to calibrate rotations in the x, y, z axis and then potential piston movements based on our joint lengths. The 3x3 portion of our Jacobian would work the same as our previous 2x2 but we would now have to get within a range of our target pending on our robots joints. Our cameras could be complicated with the new angles or if we position one over the x,y axis and one over the z axis we could simply use the incoming x,y coordinates from the first camera and only the z coordinate from the second camera.

Equipment: We would need a second camera with a view on the x, z or y, z plane. We would also need six motors instead of two as our degrees of freedom would increase from two to six (three rotational and three directional). We would also need a fully colored goal and end effector.

Lab 3
October 27 2021
Joshua Jarman
Nathan Pelletier

Setup: Our new setup would require a floor cam on a box and a more colorful bulkier robot with six joints, three rotation and three directional.

Mathematical equations: More care would be needed to calibrate the initial Jacobian in the x,y,z directions and then in its various rotations. Our Jacobian would become a 4x4 matrix to mirror the forward kinematics of a 6DOF robot. The first 3x3 would deal with ratios that would change position into rotations. The final column of the matrix would deal with piston joints in the x,y,z directions. That being said, because visual servoing is a guess and check approach if we weren't using visual servoing we would have a vastly more complicated Newton's method to deal with.