# Lab 3 Report

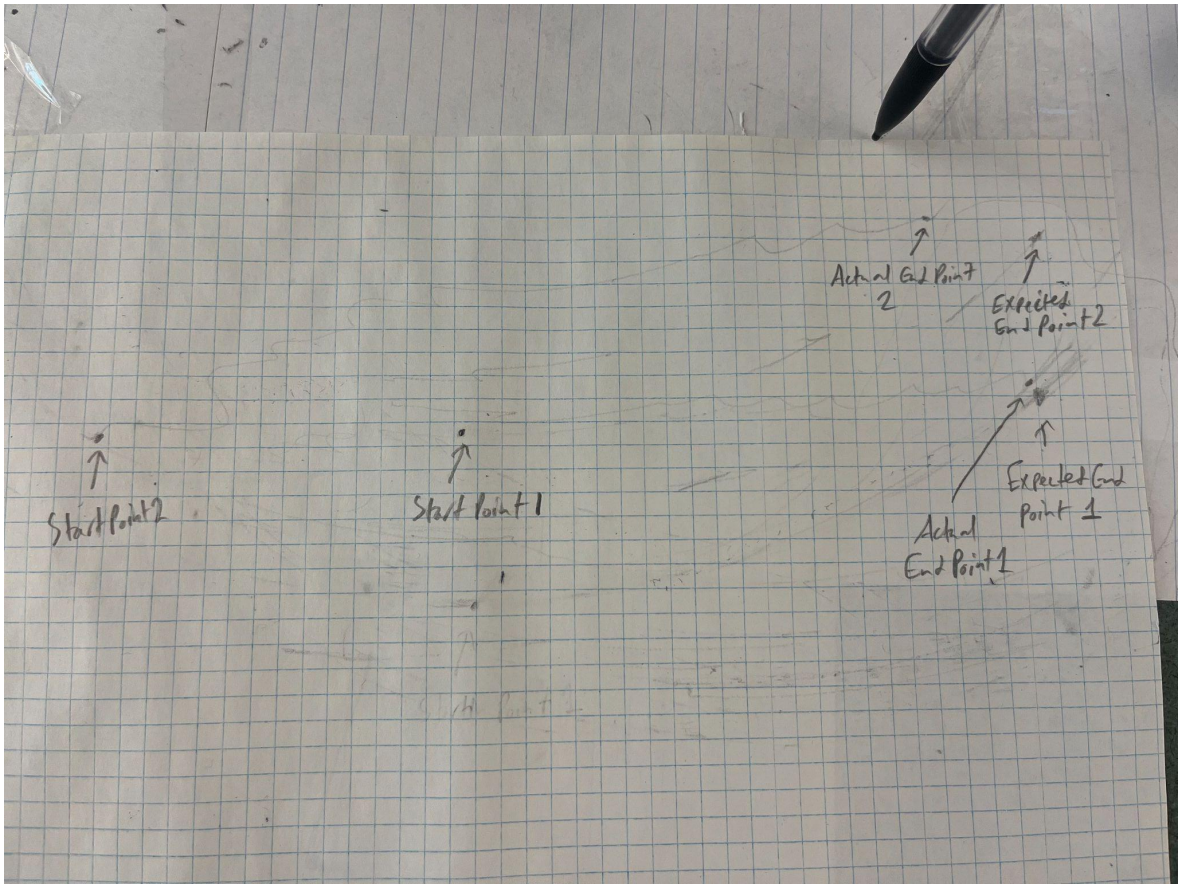
In this lab we built on concepts from lab 2 to create movement systems that could work without the arm being in a predefined starting position. To accomplish this we used both inverse kinematics, and uncalibrated visual servoing.

## Part 1:

For the first part of this lab we added the ability for the user to be able to move the end effector of our arm to two points, and then our arm would move in a straight line between the two defined points. To do this we first recorded the (x, y) locations of the end effector using the forward kinematics of our arm whenever the user pressed a button. Then once this had been done twice, we got the corresponding vector of the line between the two points. We then found 10 points along that line equal distance apart, and iteratively moved the end effector to each of those points. The method we used to move to each of these points was a modified version of the analytical inverse kinematics we implemented in Lab 2. In that implementation the choice between the arm being in the elbow up, or elbow down pose was chosen at random, this would not have worked in this scenario as we wanted the end effector to move in a straight line, and having the arm switch between elbow up and down would have prevented this. To solve this issue we choose the angle for the second joint on our arm that was closest to the previous angle of the second joint. This would ensure that if the arm started in elbow up it stays that way, and vice versa. Another issue with the analytical inverse kinematics we used in Lab 2, was that it is assumed the arm always started at the point (18, 0). This would not work for us since we wanted the arm to be able to move from any position within our workspace. To overcome this issue we treated each movement as if we were moving from the initial position, and used that to calculate the angles we needed our motor at, $\theta_d$, then we subtracted that from the angles the joints were at currently, $\theta_i$, and the difference was how much we needed to move each motor.

$$(\theta_{move} = \theta_d - \theta_i)$$

Below, you can find the path taken by our robot arm for 2 different runs. It's a bit hard to see the path taken as it was difficult to attach the pencil so that it clearly draws the path on the paper while keeping friction minimal. We have labeled the start, expected end, and actual end points for each run so it can be easier to track. As we can see, the farther the two points are from each other, the greater the accumulation of error and the farther away we end up from our expected end point.



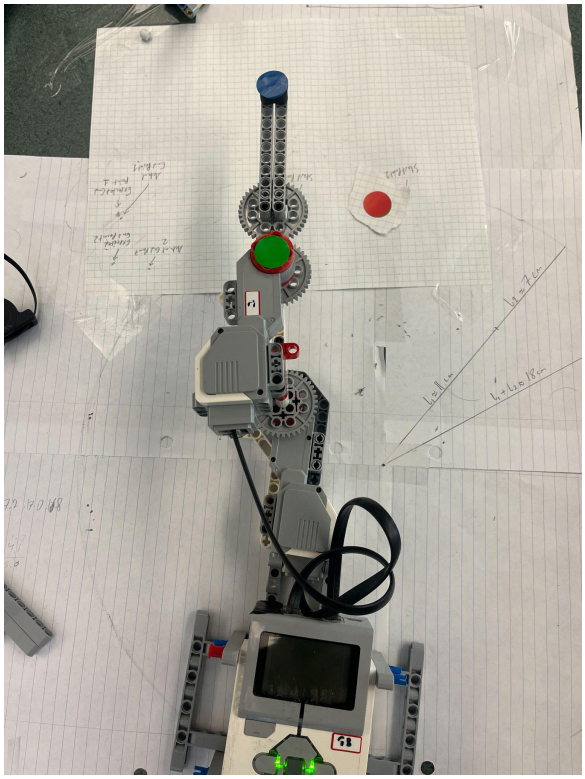

## **Part 2:**

For the next part of the lab we implemented uncalibrated visual servoing so that we could move our end effector to a goal position using just a camera, and without any knowledge of the robots kinematics. To do this we used the lab code provided to us, this took care of tracking certain colors in videos, and gave us the pixel coordinates of coloured circles. To use this we set up different coloured stickers, one on our end effector, and one within our workspace (the goal). With the position of our goal, and end effector we could then follow the steps outlined in the notes to move the end effector to the goal. The first step in this process was to create an initial approximation of the jacobian, to do this we independently moved each of the motors by 15 degrees and then dividing the respective change in pixel space by the change in joint angle see Eq. 1. Once we had our initial jacobian approximation, we could use it to move the end effector closer to the goal using the update: $\Delta\theta = \lambda * J^{-1} * error$, where error is the distance between the goal and our end effector. Once we had moved our end effector by $\Delta\theta$ we could compute a new more accurate jacobian using broyden's update every 5 movements. By repeating this process we were able to move our end effector to any goal point within our workspace.

$$J = \begin{bmatrix} \frac{x_{p1}}{\theta_1} & \frac{x_{p2}}{\theta_2} \\ \frac{y_{p1}}{\theta_1} & \frac{y_{p2}}{\theta_2} \end{bmatrix}$$

Eq. 1

Below is our setup for this part of the lab:



Robot Arm Setup

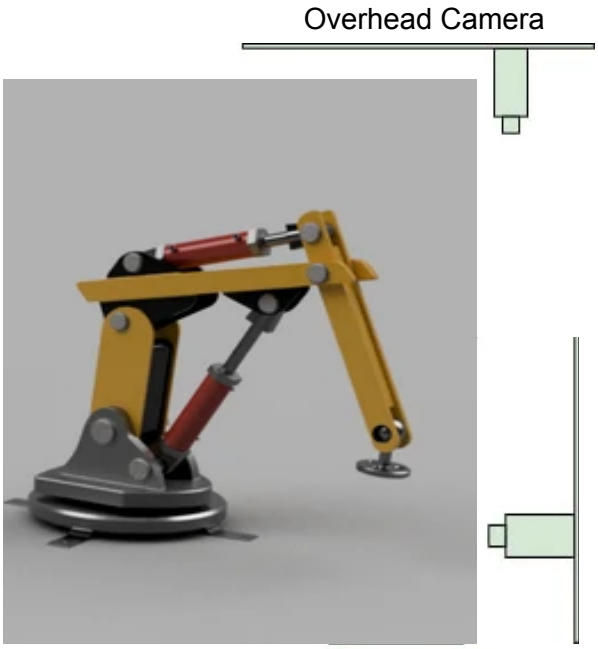
Camera Setup

You can find videos of our Uncalibrated Visual Servoing program in action here and here. If our end effector is within 25 pixels of the goal point, we end the program, as we have reached our goal. The output shown at the end is the norm of the error vector between our goal point and end effector. One issue with this setup is that depending on the position of the goal the initial estimate of the jacobian may be almost singular; a way to get over this would be to use a larger angle for the second motor when estimating the initial Jacobian.

## **Part 3:**

Lastly we would like to discuss the changes that would need to be made to our system in order to transition our implementation of uncalibrated visual servoing from the 2D coordinate system outlined above into a 3D one. The first change that would need to be made is that our arm would need to be able to move in the z direction. A simple way to do this would be to rotate our design by 90 degrees so that our arm would move in the x, z direction. Then we could add an additional motor at the base to rotate the arm and allow for it to also move in the y direction. We would also need to add an additional camera to the system to allow us to determine where the end effector and goal are in 3D space, one configuration could be to have one overhead

camera (like we do now) and one that is parallel to the table(assuming our goal is also 3D now). Finally there is the issue of blending all of this with our current program. A simple solution would be to treat each camera as a separate problem, meaning we first align the end effector with the goal in the overhead camera frame, and then once we have done that we can use the other camera to move the end effector down towards the goal. The first part of this solution would need a jacobian with 3 columns, since we would want to move all 3 motors to align the goal overhead (but only 2 rows since it can still only see in 2D). However in the second part we would keep the base rotation fixed, and only move the tow motors mounted on the arm meaning the solution to this problem would be the exact same as the one we implemented.




Camera Parallel to Table

Note: The overhead camera is to be positioned above the robot arm, pointingdownwards.