RBE 3001 Final Report

Lokesh Gangaramaney [1], Matt Schueler[2] and Arjun Gandhi[3]

***Abstract*— The final project of RBE 3001 required the use of all the knowledge gained in the previous labs, which is the using forward kinematics, inverse kinematics, differential kinematics of robotic manipulators. Robotic Manipulators are used in everyday applications from manufacturing to playing ping pong and as robotics engineers it is important for us to know the necessary knowledge required to manipulate them. In this experiment we learned how to sort objects using the RBE 3001 arm which will help us further on in our career while using robotic manipulators. This was achieved calibrating the camera in the robot and transforming the reference frame of the camera to the base frame. The coordinates from the camera are converted into joint angles of the robot with the help of inverse kinematics. The velocity of manipulator is controlled by the jacobian calculated in the previous lab and our team tabulated the gathered results in the Result section. Further we discuss the problems occured while doing the experiment and explore the knowledge gained. The lab gave us the necessary information to manipulate robotic manipulators.**

I. INTRODUCTION

The lab required the knowledge of computer vision, forward kinematics, inverse kinematics, and differential kinematics to sort objects based on the color and size of the object. This knowledge gained in this lab and the previous labs will be helpful later on in our career when we manipulate industrial arms. Arms like the Kuka, can be manipulated by using the DH convention used to set up the reference frames of the RBE 3001 arm. Using the knowledge of computer vision, our team calibrated the camera and set-up the intrinsic and extrinsic parameters of the camera object. After calibration, our team enhanced the images to segment the edges of the objects to get all the information required to sort the objects. However, since the reference frame of the camera and the reference frame of base of the robot are not the same our team used forward kinematics to transform the reference frame from the camera to the base frame. Our team used inverse kinematics to get the joint angles of each joint to orient the RBE 3001 arm according to x,y,z coordinates of the sorting objects with respect to the base frame. These coordinates are determined by segmented images processed by the camera and transforming the coordinates from the camera to the base frame. Our team used the inverse kinematics calculations did in Lab 3 to get the joint angles for the robot while sorting objects using the geometrical method. Finally, our team used the jacobian calculated in Lab 4 to control the velocity of the robots links to sort the objects. The methods and calculations required to do the mentioned experiment is explored further in the Methodology section. Using the knowledge of inverse, forward and differential kinematics the team was able to manipulate the RBE 3001 arm to sort the necessary objects and tabulated the gathered results in the Result section of the report. Further we discuss the problems we had while doing the experiment and summarize all the knowledge gained in the experiment.

*RBE 3001 C'19 Organization

[1] Lokesh Gangaramaney. Gangaramaney is with Student Body of Robotics Engineering, Worcester Polytechnic Institute, Worcester Polytechnic Institute, 100 Institute, Worcester MA, USA

[2] Matt Schueler. Schueler is with Student Body of Robotics Engineering, Worcester Polytechnic Institute, Worcester Polytechnic Institute, 100 Institute, Worcester MA, USA

[3] Arjun Gandhi. Gandhi is with Student Body of Robotics Engineering, Worcester Polytechnic Institute, Worcester Polytechnic Institute, 100 Institute, Worcester MA, USA

II. METHODOLOGY

A. Positional Kinematics

1) Forward Kinematics: DH Reference Frames: Our team set their arm to position where they could assign DH frames to the joints (Shown in Figure 1) and assigned the DH frames according to the convention. The first joint rotated counter-clockwise, therefore by the right hand rule the z0-axis points in the upwards direction as seen in Figure 1. The x0 and y0 reference axis are assigned by the right hand rule using the Z-axis. In the next rotational joint, the rotation was counterclockwise in the orientation seen in Figure 1. Therefore we got the z-axis shown, and since z0 and z1 intersected each other the common normal was perpendicular to the plane formed by them. Therefore, the x1-axis was selected to be as shown to make calculations easier. The y1 axis was found using the right hand rule. The next rotational joint had the same rotation as the previous one, therefore the z2-axis was similar to the z1-axis. However, since they are parallel the x2-axis was perpendicular to z1 and z2 axis, in the same plane as x1. Y2 axis was formed using the right hand rule. The x3, y3, and z3 axis are made to simplify the calculations for the arm. Z3 is parallel to z2, x3 is along the common normal of z2 and z3 in the same plane as x2, and y3 was found from the right hand rule.

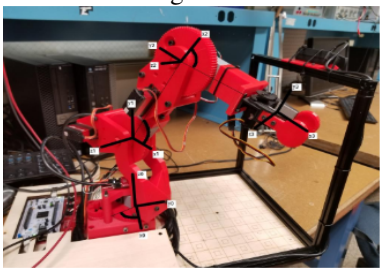


Figure 1: DH Reference Frames of the RBE 3001 Arm

2) Forward Kinematics : DH Table:

After the DH frames were checked off by a teaching staff member the team used the DH convention to fill the table of DH parameters. For the first link, the rotation about the z-axis is defined by joint angle in the first rotation, therefore the theta for first link is theta1. Similarly the thetas for link 2 and link 3 are theta2 and theta3 respectively. From link 1 to link 2, the frame rotates pi/2 with respect to x0 and therefore the alpha for link is pi/2. Since the z-orientation is the same for links 1, 2, and 3, the alpha for links 2 and 3 is 0. The new center for reference frame 1 moves 135 mm in the z-axis, therefore the d is 135mm for link 1. However there is no movement along the z-axis for links 2 and 3, therefore the d is 0 for links 2 and 3. There is no translation along x0 axis in link 1, therefore the a is 0 for link 1. However there is translation for links 2 and 3 in the x-axis, therefore the lengths are 175 mm and 169.28 mm for a in links 2 and 3 respectively. The table of parameters can be found in Table 1, DH Parameters table

DH Parameters				
Link	θ	α	d	a
1	θ_1	$\pi/2$	135	0
2	θ_2	0	0	175
3	θ_3	0	0	169.28

B. Inverse Kinematics

For the inverse kinematics calculations the team assigned the necessary coordinate frames at the joints of the robot using the Denavit-Hartenberg convention. The first joint has a rotation in the base axes therefore the z-axes comes upward according to the right hand rule, the x and y frames are found using the right hand rule as seen in Figure 1. This coordinate frame is used as reference for the geometry required to find the joint angles. Therefore the first joint angle (theta 1) is defined as the inverse tangent of the y-component of the tip position over the x-component of the tip position. The projection of the tip position vector onto the XY plane is labelled as n, which can then be used to define a number of other distances. The length l is defined as the hypotenuse of the triangle formed by (Pz-L3) and n. This l then forms a triangle with L2 and L3. Using the law of cosines on this triangle we get that the square of the third link is equal to the sum of square of the second link and L subtracted by twice the product of second link, link L and cosine of beta angle formed in the triangle. Doing some algebra we find that the beta angle is equal to inverse cosine of the sum of square of link 2 and L subtracted by the square of link 3 over twice the product of L2 and L. Theta 2 is the sum of alpha and beta, where alpha is given by the inverse tangent of the z-component of p subtracted by first link length over n. If we look at the triangle again we find that the using cosine formula we can find theta 3 by the same procedure described above.

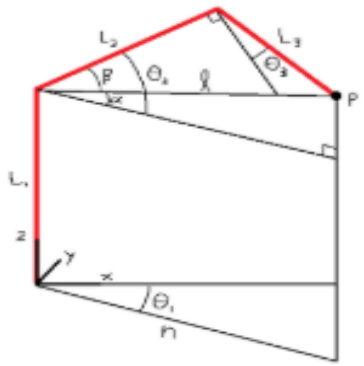


Figure 2: Geometrical Respresentation of the RBE 3001 Arm

$$\theta_1 = arctan((\mathrm{p}_x / \ p_y))$$
$$\mathrm{n} = \sqrt{\mathrm{p}_x^2 + p_y^2}$$
$$\mathrm{l} = \sqrt{(\mathrm{p}_z - L_1)^2 + n^2}$$
$$L_3^2 = L_2^2 + l^2 - 2L_2 l cos(\beta)$$
$$L_3^2 - L_2^2 - l^2 = -2L_2 l cos(\beta)$$
$$\beta = arcos(\tfrac{L_3^2 - L_2^2 - l^2}{2L_2 l})$$
$$\theta_2 = \alpha + \beta$$
$$l^2 = L_2^2 + L_3^2 - 2L_2 L_3 cos(\pi/2 + \theta_3)$$
$$l^2 = L_2^2 + L_3^2 - 2L_2 L_3 cos(theta_3)$$
$$\theta_3 = arcsin(\tfrac{l^2 - L_2^2 - L_3^2}{2L_2 L_3})$$

Inverse Kinematics Calculations

Using the formulas above the team put them in MATLAB to get the joint angles from the x,y, and z position. With a conversion formula the team converted the joint angles to encoder values to control and manipulate the RBE 3001 Arm using inverse kinematics.

C. Velocity Kinematics

The experiment required the team to calculate the Jacobian of the RBE 3001 3 DOF Arm, a matrix that maps a 3*1 matrix vector of joint space velocities to a 6*1 vector of the end effector linear and angular velocities, that is:

$$\mathrm{p}^{\cdot} = \mathrm{J(q)q}^{\cdot}$$

The Jacobian is calculated by the column method as shown in figure 3. Column method uses transformation matrices to calculate the Jacobian. Therefore, we wrote down the transformation matrices using the DH parameters we determined in Lab 2. We calculated the transformation matrix for $T^1{}_0$, $T^2{}_0$, and $T^3{}_0$.

Different Z_is were found for $T^1{}_0$, $T^2{}_0$, and $T^3{}_0$ and were used to multiply the results found from finding (pe - pi). We got results for J1, J2 and J3 from the result of those multiplications and got the Jacobian J.

$$\mathrm{J(q)} = [J_1(q_1)\ J_2(q_2)\ J_3(q_3)\ .....\ J_n(q_n\ ]$$
$$\mathrm{J(q)} = [z_i\ \mathrm{x}\ (p_e\ -\ p_0^i)]$$

Jacobian Formula

$$\begin{bmatrix} 169.28 + cos(theta_3) + sin(theta_1) & 0 & 0 \\ 169.28 + sin(theta_3) + sin(theta_1) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

*

Jacobian

Using the Jacobian calculated above, the team wrote a function in MATLAB, jacob0, that takes an input q (the joint angles of our configuration) and returns the corresponding 6*3 matrix. To validate the Jacobian, the team passed the robot an overhead configuration of the arm fully extended along the Z0 axis (Singularity) and calculated out the determinant of Jp (Position Jacobian). The team found that the function worked correctly as the first column of the Jacobian contained all zeros and the determinant of the Jacobian came out to be zero.

The team then wrote a function in MATLAB to calculate the linear velocity of the end effector by solving for

$$\dot{p} = J(q)\dot{q}$$

The function takes the current joint angles q, and the instantaneous joint velocities $\dot{q}$ as arguments, and returns the task space velocities $\dot{p}$.

Next the team selected three arbitrary points in the task space that required motion of all three axes to travel between them and interpolated between each pair of points using a cubic polynomial trajectory. We then used the inverse kinematics functions developed in the previous lab to convert these points into joint space. The team reused the STATUS command to continually read the joint angles and velocities and used this data to continuously calculate the linear velocity vector from the Jacobian. The live stick plot displayed the velocity vector at the tip of the end effector with its length proportional to the magnitude of the velocity.

The team used the plotellipse function provided on Canvas by the teaching staff and configured the live stick model plot of the robot developed in prior labs to show the manipulability ellipsoid of the RBE 3001 arm. The plotellipse function takes as input a square positive-definite matrix A, and plots the three-dimensional ellipse described by $x^T A^{-1}$x.

Without having the power on, the team manually dragged the arm around with the plot on, and discovered singularities by looking at the manipulability ellipsoid. The team identified two singularities and included them in the results section of the report. To stop the robot from reaching these singularities, the team implemented a function to stop the robot from having singular configurations. This was done calculating the determinant of the Jacobian constantly and checking if the determinant of the matrix is close to zero and right before where the Jacobian reached zero the robot has stopped.

The above experiment was repeated for the inverse kinematics solution of the RBE 3001 Arm. The team validated their algorithm by implementing the following tests:

To test the functionality of the differential kinematics functions written in the lab, the team created a graphical interface to the robot simulation using the ginput function. When an arbitrary position is clicked in the XZ plane, the code will attempt to iteratively approach this point until it is sufficiently close. This was done for 3 points, which were then compared to the solutions found by our forward kinematics functions.

D. TRAJECTORY GENERATION

The team moved the robot in 3 arbitrary positions and recorded the joint encoder values for them. The robot was then told to move to these 3 positions, and the joint data was recorded along with the timestamps to allow the team to complete the final part of the lab that involved waypoint-based navigation through space.

Using Formula 1 below, the team was able to create a function that takes in desired start and end times t0 and tf (in seconds), start and end velocities (in encoder ticks/sec), and start and end joint positions (in encoder ticks) and outputs a 4x1 array containing the coefficients a0 through a3 for a quintic polynomial in MATLAB.

$$\begin{bmatrix} -1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}$$

Quintic Polynomial System of Equation

This function was used to create the same motion trajectory created earlier in the lab. 3 arrays were then created by plugging in 10 points evenly spaced along the curve (in the time axis) to create multiple additional waypoints to travel to. The resulting trajectory traced the same triangular path as before.

This triangle was linearly interpolated with 10 evenly spaced points between each pair of triangle. This was simulated in real time in MATLAB and three plots of position, velocity, and acceleration for x,y and z axes.

Creating a quintic polynomial the team repeated the above step to interpolate the three points of the triangle. After creating these plots, the team compared them with the plots generated in the previous step.

The results generated are presented in a professional manner in the results section of the report.

E. COMPUTER VISION

The first thing we did in this lab was test the connection between the robot and the webcam. First, we used the webcamlist() function to see if the firmware was able to see the camera. Then using the webcam() and preview() functions, we were able to get a live feed from the camera. Since we were able to get images from the camera, we could move on to calibrating the camera to calculate its parameters.

To do the camera calibration, we used the camera calibration app from the Computer Vision System toolbox. Using the camera feed from the robot, we captured a large number of images to ensure that the software would have enough images to calculate the parameters of the camera, even if there were some rejections. After capturing all the images, the software ran calibration for the first time. We were able to get 42 images where the software was able to detect the checkerboard, however some of the images had a significantly higher reprojection error, so we manually rejected some of them. By the end of this process, we had 37 images that were then used to generate the camera parameters object to be used by the rest of the code. The histogram of this process can be seen in Figure 4 below. In figure 5 you can also see the placements of the checkerboard in each image that the camera was able to detect.

However, the camera reference frame is not enough to figure out where objects are in relation to the robot. To be able to manipulate objects in the workspace, we need the transformation matrix between the robot base joint and the camera itself. Using the function getCamToCheckerboard provided in the github repository, we calculated the reference frame of the checkerboard relative to the camera. Then, using measurements given by the lab document and the size of the checkerboard, we calculated the reference frame of the checkerboard relative to the base joint, shown below.

$$\begin{bmatrix} -1 & 0 & 0 & -52.8 \\ 0 & 1 & 0 & 276.6 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using these two reference frames, we wrote a MATLAB script to calculate the transformation of the camera relative to the base joint. This transformation matrix is shown below.

$$\begin{bmatrix} 0.0272 & -0.9961 & 0.0845 & 0.0857 \\ -0.9485 & 0.0010 & 0.3168 & 307.7496 \\ -0.3157 & -0.0888 & -0.9447 & 290.2403 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

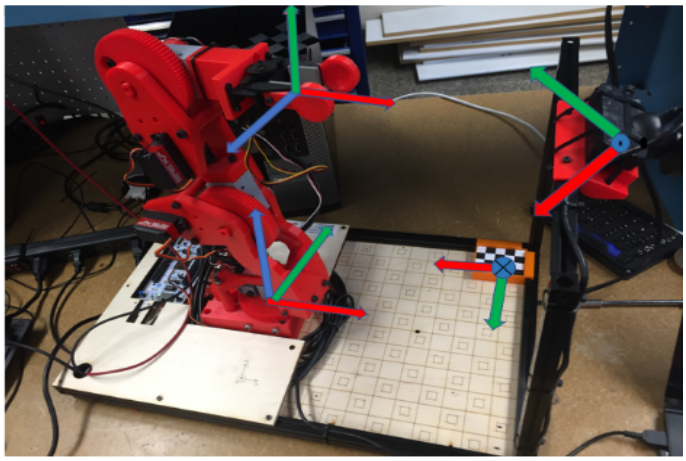


Figure 5: Camera to base frame transformation, Blue is X ; Green is Y ; Blue is Z.

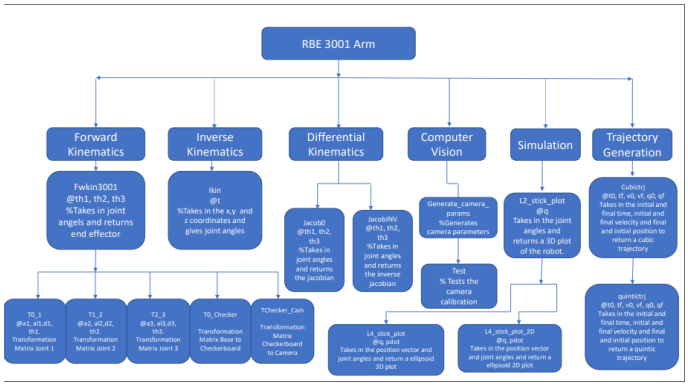
To verify the transformations, the team took a picture of the workspace and identified 4 points in the robots reference frames whose coordinate were known. Read the pixels coordinates of these points in the camera image. By using the pointstoWorld function in MATLAB our team converted pixel coordinates to points in the checkerboards reference frame.Converting the coordinates from the checkerboards reference frame using the transformation matrix described above.

Moreover, to get the centroids of the solid colored spherical objects our team used the digital imaging processes methods covered in class and implemented the required function. Further we converted these centroids to coordinates with respect to the robots reference frame. Building on the previous step our team wrote code to recognize the objects color and size. There were objects of three different colors (blue, yellow, and green) and two different sizes and got the required sign-off.

To sort the objects we wrote a C++ script to control the servo-actuated gripper. This was achieved writing a simple PWM signal learned from previous classes.

To implement the sorting algorithm we wrote a script that can determine the 3D position of the object with respect to the robots reference frame. We calculated the inverse kinematics from the previous labs to figure out the joint angles required to the robots end effector to reach the objects location and sort the objects based on color and size.

The Functional API Library used for the experiment is shown in the next page.



UMLDiagram

III. RESULTS

flushleft Our team got the final results for the final lab of the course :-

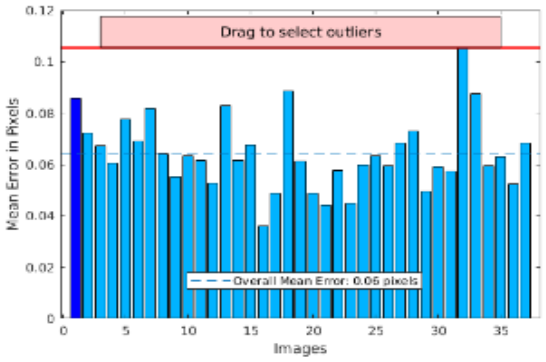


Figure 3: Reprojection Errors of Camera Calibration

flushleft The above histogram displays the rejection of MATLAB's camera caliberation algorithm while getting accurate camera parameters to sort the objects accodring to their base and color

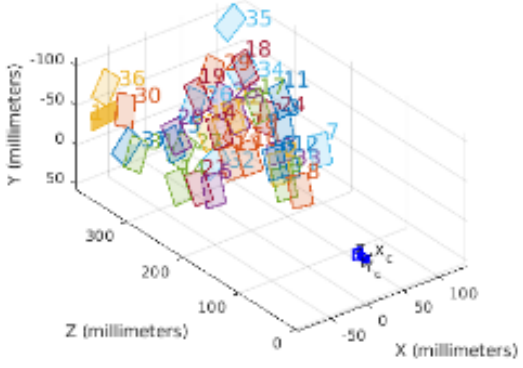


Figure 4: Locations of Checkerboard in Calibration Images

flushleft The different locations used to caliberate the camera are shown in the 3-Dimensional plot above. Our team used various arbitrary locations to get the best camera calibration possible for the robot. This camera calibration proved useful further on as the sorting algorithm was accurate with the perfectly caliberated camera.

IV. DISCUSSION

The lab started out smoothly. Our team was able to calibrate the camera during the lab time. We got the necessary parameters and were able to verify that the parameters were accurate enough for the sorting algorithm. Using these parameters to image process the different orbs proved more challenging than we thought. But we were able to identify objects correctly enough to sort them according to their color and radius. Adding gripper control using C++ was done by creating a new server ID to take from MATLAB and our team modifying the firmware to open and close the gripper from MATLAB. However after this was tested and checked we had problems trying to open and close the gripper. It turned out to be a hardware problem, since the gripper's wire came out as the robot tried to extend its arm. We did not check that at first and tried to debug the C++ code, our team lost a lot of time because of that. After, we solved that issue we tried to implement the sorting algorithm, but the robot's shoulder joint had troubles implementing our code. It broke down after constantly testing our code. We had to sit down with a teaching staff SA and fix the robot's joints. Our team lost a lot of time during the repair. Even after the repair the team found that the motors for the joint were burned out. Thus to advance further in the lab our team used Team 1's robot with the teaching staff's permission. Since the robot's camera was different, we had to calibrate our camera again. After repeating the progress we had already made we found that the forward kinematics, inverse kinematics code we wrote in the previous labs for the original robot did not work well with the new robot. We had to calibrate our home position, encoder values again. Even after the significant challenges our team was able to pickup objects with PID control, but we found that the objects would generally shoot out of the robot's gripper as the PID control had a lot of jerk. We tried to implement a linear interpolation and a quintic trajectory function. But implementing the quintic function messed up our entire sorting algorithm. We debugged this issue for hours, trying to find any bug we could have made but found our function to be correct. Moreover we could not get any teaching staff's help during that time as our team was behind than most teams, and the issue came up after most teams had completed the challenge. We recognize that the teaching staff had office hours after the project-presentation, but we could not incorporate them into our limited time. Finally, the team was able to dynamically track the objects in real time and got half points for the final demo. Even still we were trying to get the plots for the trajectory generation and could not include the plots in the report as the new arm broke again while filming the video for the robot.

V. CONCLUSION

We learned important robot theory regarding the denavit-hartenberg convention, forward kinematics of a robot, inverse kinematics, velocity kinematics, computer vision and path planning. We implemented a MATLAB script to demonstrate the knowledge of the material taught to sort objects according to their color and size while using the RBE3001 Industrial Arm. We found that the script we wrote worked well for the forward kinematics, inverse kinematics, computer vision. However our path planning function was off, therefore, moving further we will try to improve our trajectory generation for Industrial arms.

ACKNOWLEDGMENT

.

Our team acknowledges the contributions of Professor Fichera, the teaching staff, Kevin Harrington, Alex Camilo for the firmware provided and the assembly of the RBE 3001 Arm.

REFERENCES

Video for the completition of the Lab. https://drive.google.com/file/d/1XiIFTS4zJFbh2b5sRK4RRO$_m BN9StCXB/view?usp = sharing$