Automated Sorting System: Final Lab Report

Nicholas Lanotte
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts, United States
njlanotte@wpi.edu

Max Luu
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts, United States
mluu@wpi.edu

Karina Naras
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, Massachusetts, United States
krnaras@wpi.edu

***Abstract*—The goal of this lab was to understand how the individual elements that have been developed throughout this course can be combined to form a fully autonomous system. The automated sorting system required image processing, force sensing, inverse kinematics, and Jacobian mathematics to locate objects in the task space, move the end effector to an object, pick it up, weigh it, and place it in a specified location based on color and weight. The final system tracked multiple objects in a workspace, targets objects based on a color priority (blue, green, yellow), then sorts them either to the left or right based on weight and color.**

Keywords—manipulation, automation, sensing, computer vision, kinematics

I. Introduction

This lab was intended to combine all the skills developed during throughout the term to generate a fully autonomous robotic sorting system. Using a camera, the robot located objects in its workspace, moved the end effector to pick them up, determined which weight category they fit into, and released them in an area categorized by their physical traits.

The torque sensing, which was used to differentiate between objects with heavy or light bases, required use of the strain gauges that are located at each of the robot's joints. The output values from the sensors had to be calibrated to convert the ADC readings to torque values. The Jacobian and torque sensor readings were used to calculate the force vector of the end effector, which could be used to determine whether the object being held by the end effector has a copper (heavy) or plastic (light) base.

The camera mounted over the task space provided visuals that could be used to track objects in the workspace. The images were segmented based on the three colors that target objects could be, and the processed images were used to find the centroids of the objects in the task space. The segmentation functions also noted the color of the object being picked up, so that the objects could be sorted both by weight and color. The centroids of the objects were used as desired positions for the end effector, and functions were written to use the gripper to pick up the objects and place them outside the task space

II. Methodology

A. Force Sensing

The robot arm used a strain gauge at each joint to sense torque. The packets sent in response to the STATUS command included joint angles, joint velocities, and joint torques (the values were expressed in raw ADC values; this will be addressed below). First, the sensor readings needed to be averaged in order to obtain smooth torque signals. This was done in the Nucleo firmware by constantly polling ADC readings and averaging the last 5 of them. Whenever a status packet was requested, the last averaged value was returned. This was done in the Nucelo firmware rather than Matlab because averaging 5 readings in Matlab would have taken about 0.1 seconds and require 5 packets to be exchanged to and from the robot. The Nucleo can average 5 values in microseconds and keep an updated reading ready to be sent.

In order to convert the ADC counts into Newton-millimeters, Equation (1) was implemented with y as the raw ADC counts, τ as the applied torque in Newton-meters, k as the scaling factor equal to 1/178500, and y_0 as an offset.

$$y = k*\tau + y_0 \tag{1}$$

k was a specified constant that allowed the robot to get readable torque values. These values may not have been accurate but they were sufficient to progress further into the assignment. The k value was planned to be tuned after the other parts of the lab were functional. This equation was algebraically manipulated so that the torque values could be isolated

$$\tau = (y - y_0) / k \tag{2}$$

The offset was determined by configuring the robot in an upward singularity where all of the torques on the joints would be zero. The raw ADC values were read in and were entered as offsets for each corresponding joint in the Matlab code.

The torque sensor readings were used to calculate the force applied at the end effector, which is how the weight of the objects picked up is determined. Equation (3) for the joint torques and task space force was manipulated into Equation (4) as seen below

$$\tau = J_p^T(q) * F_{tip} \;\rightarrow\; F_{tip} = \tau * J_p^T(q)^{-1} \tag{3, 4}$$

The Jacobian was the same as was calculated in previous labs, and the inverse could be taken easily with Matlab. The code previously used for live plots of velocity was modified to instead display force vectors on the end effector.

B. *Vision Based Tracking*

The Sony PlayStation 3 camera was added to the system on a stand overlooking the robot's task space. It could display a live video feed in Matlab from which snapshots could be taken, and processing those images allows detection of objects in the workspace. The provided calibrate_camera.m script was used to implement a point-based registration algorithm which was used to map the relationship between the camera's field of view and the reference frame of the robot. This calibration process generated a file called pixel.xml which contained the specific numbers needed to map the task space to reference frame; this file was used with the given mn2xy function to transform coordinates between the camera's reference frame and the robot task space.

The Color Thresholder app in Matlab was used to segment images for each color object; the HSV color space was chosen because it was least likely to be affected by changes in environment lighting. The team made good use of the Export Function tool, and made one function for each color, yellowOrbs, greenOrbs, and blueOrbs. The segmented black and white images outputted by these functions were inputs for the function findObject which outputs the color of the spheres it sees, and made red "+" markers on video feed from the camera for each centroid. The centroids of all the visible objects were stored in an array with three columns: the first column contained either a 1, 2, or 3 to indicate color, and the second and third columns are the coordinates of the centroids. These coordinates were used to navigate the end effector to positions above objects in order to pick them up.

C. *Automated Sorting*

The automated sorting system was comprised of two parts that run sequentially. The first part was responsible for tracking objects in the workspace and moving the arm to into position to pick up the objects, while the second part lowered the arm, grabbed objects, and sorted them. First, the workspace was checked for objects with centroids (segmented portions of the camera view with a large enough area to be objects). The program eliminated centroids that, though in view of the camera, were not in the workspace; this prevented the robot from picking up objects that it had already sorted. After removing irrelevant centroids, it checked again to determine if any objects had moved. One object at a time was targeted, and the robot moved to a position above it. Checking for object movement frequently prevented the robot from having to complete trajectories that were no longer leading to an object, and it could instead create a new trajectory. Once the robot reached a position above an object, it entered into the second section of sorting code.

During development, steps were taken to slow the overall movement of the robot so that when the end effector descended to reach an object, the cups on the gripper would not move so fast that they bump the object out of the way. The robot descended over an object, closed the cups around the orb, lifted it up slightly, and paused to weigh the object. Based on the color and weight of the objects, the robot calculated a trajectory to the designated location for that object type, moved there, and dropped the object where it belonged. Once the sort location was reached, the code resets to target another object in the workspace, and the cycle was repeated. A flowchart of the sorting system code can be found in the Specifications Document submitted previously.

III. RESULTS

The live plot for the force sensor was successful in plotting the given force in the given configuration. Forces were applied to the end effector by hand as seen in Figure 1. This test revealed that the strain gauges were not returning accurate values though. For large forces the force vector direction was accurate, shown in Figure 2. After releasing this force from the robot, the end effector force stayed constant, making it apparent that the arm is deforming, which created permanent stress on the strain gauge. The function created was able to print the resulting force in its separate components, the magnitude and the unit vector representing direction as seen in Figure 3.

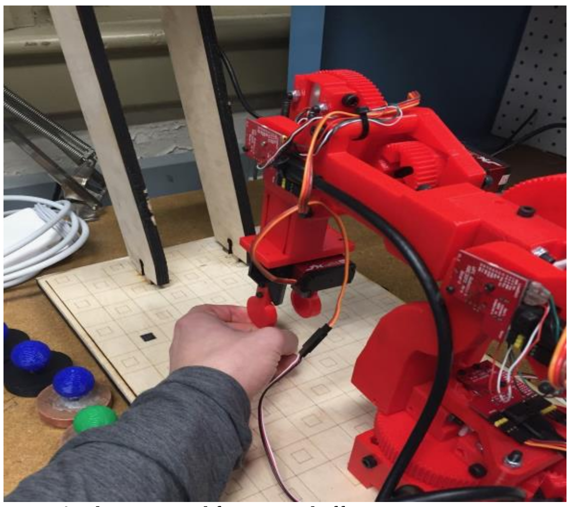


Fig. 1 Applying upward force to end effector

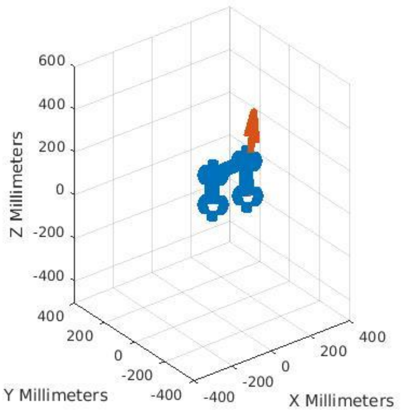
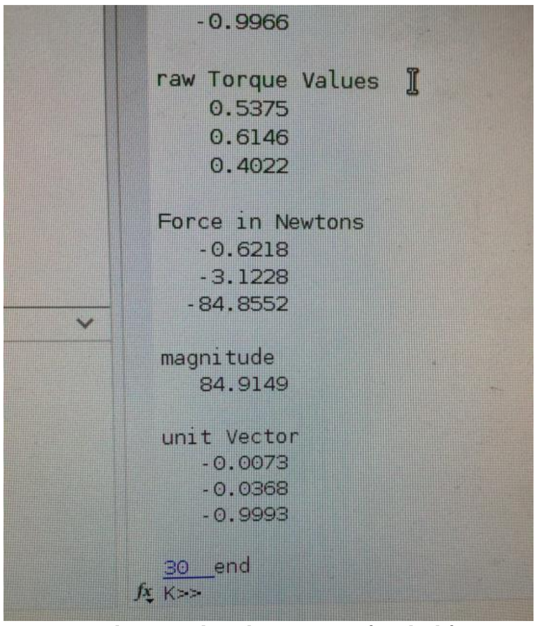


Fig. 2 Live plot of force vector from applied upward force



```
  -0.9966

raw Torque Values
    0.5375
    0.6146
    0.4022

Force in Newtons
   -0.6218
   -3.1228
  -84.8552

magnitude
   84.9149

unit Vector
   -0.0073
   -0.0368
   -0.9993

30   end
K>>
```

Fig. 3 Printed magnitude and unit vector of applied force

The following figures depict the three tests conducted to verify that the arm could reach to different object locations correctly. These are shown in Figures 4, 5, and 6.

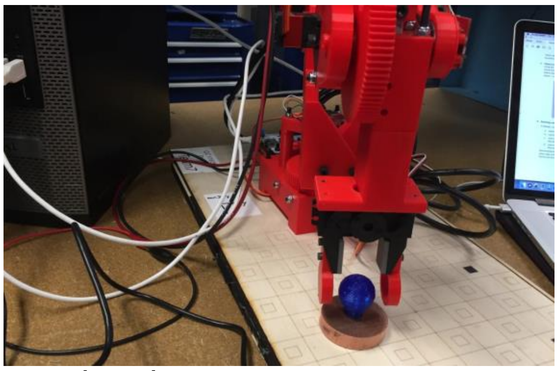


Fig. 4 Robot reaching test 1

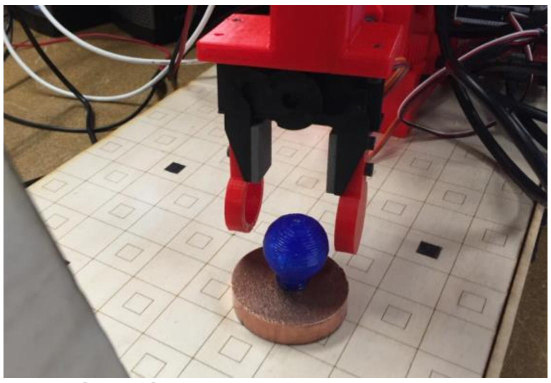


Fig. 5 Robot reaching test 2

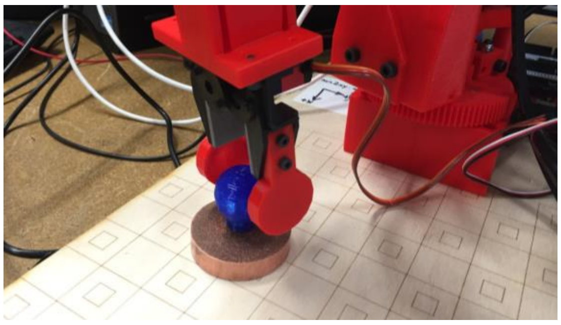


Fig. 6 Robot reaching test 3

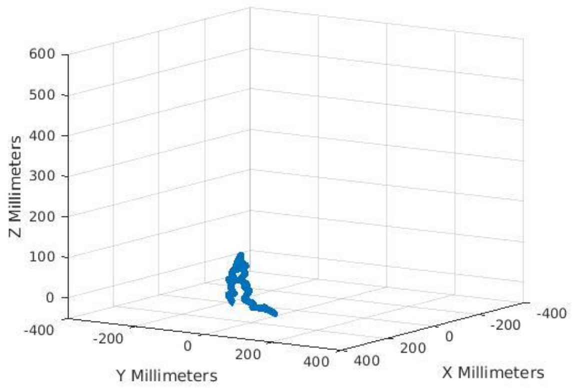
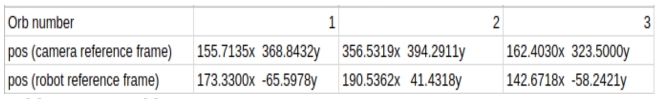


Fig. 7 Robot trajectory to pick up an object

Figure 7 depicts the trajectory followed by the arm when reaching for an object. First it follows a cubic trajectory up over the object, then another cubic trajectory down to reach for the object.

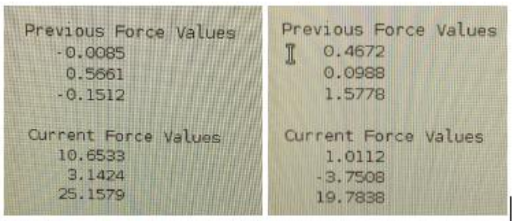
The robot had no issues generating trajectories to reach for the orbs. We used a cubic polynomial to generate these trajectories. It would first reach up to avoid other orbs in the workspace, then reach above the target orb it was tracking. The data in Table 1 shows the centroid positions that were reached for in reference frame of the camera and also the reference frame of the robot.

Vision based tracking, automated sorting by color, and even dynamic camera tracking were successfully implemented in this lab. The robot was able to consistently locate objects in its workspace, pick them up, and sort them based on color. Sorted objects were deposited in designated locations: from the “point of view” of the arm, objects with heavy bases went the left of the workspace platform, objects with light bases to the right, with yellow objects being placed nearest to the robot, then green objects, then blue objects farthest away. The robot rarely fails to grab objects due to bumping them out of the way, and it efficiently changes paths when the object it is targeting is moved. The only part of the system that does not consistently work is the weighing of objects, which is the result of faulty strain gauge assemblies. A light object was measured first resulting in the output shown on the left of Figure 8. Then a heavy object was measured in the same sorting run and is shown on the right of Figure 8.



Orb number	1	2	3
pos (camera reference frame)	155.7135x 368.8432y	356.5319x 394.2911y	162.4030x 323.5000y
pos (robot reference frame)	173.3300x -65.5978y	190.5362x 41.4318y	142.6718x -58.2421y

Table 1 Centroid locations



```
Previous Force Values
   -0.0085
    0.5661
   -0.1512

Current Force Values
   10.6533
    3.1424
   25.1579
```

```
Previous Force Values
    0.4672
    0.0988
    1.5778

Current Force Values
    1.0112
   -3.7508
   19.7838
```

Fig. 8 Force measurements for light and heavy objects

IV. Discussion

The force sensors on the joints of the robot are strain gauges that fit inside designated spaces in the 3D-printed arm links. The strain gauges themselves are very sensitive to any force applied to them, which is useful in applications like this where light objects are being lifted. However, the joint structures seemed to be deforming with stress, causing permanent forces to be applied to the strain gauge. Part of the process of determining why the force sensing wasn't working despite a logical code approach was a test to detect deformation in the system. The arm was raised vertically to an equilibrium position, and the voltage across the strain gauge in joint 1 was measured with an oscilloscope. The voltage reading was 2.05V shown in Figure 9.

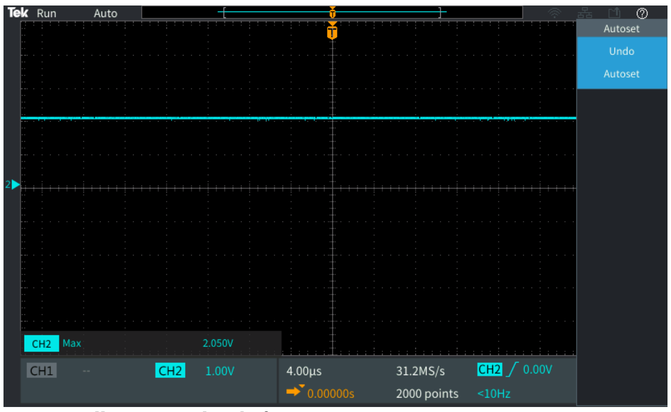
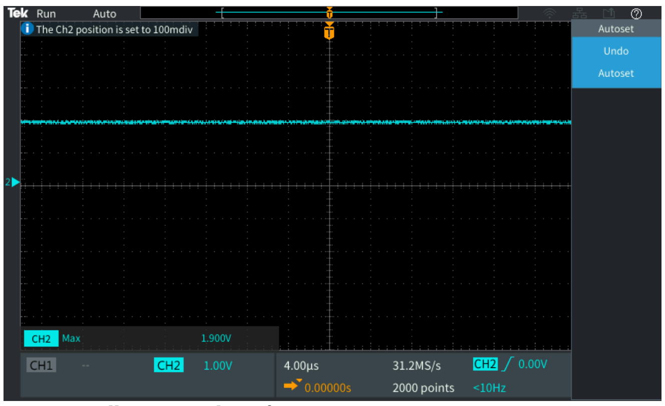


Fig. 9 Oscilloscope reading before strain test



The Ch2 position is set to 100mdiv

Fig. 10 Oscilloscope reading after strain test

The arm was then lowered, and a strong force was applied to the end effector in the direction of rotation of joint 1 of the arm by hand. When the arm was raised vertically again, the voltage reading was 1.9V shown in Figure 10, which is a very large difference for this particular sensor. Logically, the strain gauge readings should have been the same any time the robot was in that equilibrium position. The fact that the readings changed means that deformation in the system is applying pressure to the gauges, so they are constantly receiving readings that make small changes in force, such as those required to determine the difference between objects' weights, difficult to detect. The constantly changing forces within the arm system make getting force readings consistent enough to determine the difference between a heavy, light and no object effectively impossible.

The vision sensing was successfully implemented but in certain areas of the workspace the camera would determine centroids to be slightly off their actual position. The algorithm that converts the camera position to coordinates in the robot's reference frame assumes that there conversion factor is the same for all positions in the camera view, but due to the angle of the camera, positions towards the back of the reference frame appear to be more inward towards the y axis than positions closer to the camera. This was not accounted for resulting in the robot having trouble reaching for objects near its base but this problem could be fixed in software if more time was given.

V. Conclusion

This project was a valuable combination of concepts learned throughout the course. Previous live plotting programs were developed further to incorporate force vectors, and the inverse kinematics and Jacobians were crucial to the robot's ability to move to objects. These elements of automated arms have many practical applications that may be encountered in the workforce, particularly in industrial robotics. The sorting function of the 3-DOF manipulator is something that may be implemented in further robotics applications. Understanding the fundamentals of computer vision, D-H parameters, and Jacobian kinematics utilized in this lab will provide a leading edge in future robotics work.

Acknowledgment

We would like to thank Dr. Loris Fichera for teaching this course on robot automation, and Kevin Harrington for providing the lab materials used. We would also like to thank Nathaniel Goldfarb, Vinayak Jagtap, Keion Bisland, Kacper Puczydlowski, and Duong Nguyen, the Teaching Assistants and Student Assistants who have helped us through difficulties in the lab.