Robot Pick And Place System

WPI RBE3001 C23 Team 3 Final Report

Calvin Page
Worcester Polytechnic Institute

Arturo Lemos Lee
Worcester Polytechnic Institute

Puen Xu
Worcester Polytechnic Institute

***Abstract*—In order to increase efficiency and reduce labor cost, we developed an autonomous robot pick and place system which consist of a 3 degree-of-freedom spherical robot arm, a camera, and a robot claw that would constantly identify the location of a ball, move to the location upon quintic trajectory, pick up the ball, and drop it to different positions based on its color. We used position and differential kinematics along with motion planning to ensure the robot arm follow a smooth trajectory.**

***Index Terms*—robot manipulation, computer vision, motion planning**

I. Introduction

An autonomous robot pick and place system is very useful in areas such as packaging, welding, and waste recycling. Not only that it cuts down labor cost, it increases efficiency in a variety of industries. Here we present a robot pick and place system with a 3 degree-of-freedom spherical robot arm that integrates robot manipulation, motion planning, and computer vision to sort balls of different colors. We will first describe the development and implementation of the robot system, including the derivation of the position and velocity kinematics, system architecture, robot arm model, and the image processing pipeline. Then we illustrate and analyze the results we obtained such as the intrinsic and extrinsic camera calibration, methodology of object detection and localization, and the logic of the sorting system. Finally we conclude the report with comments about the current project and suggestions for future projects.

II. Development and Implementation

A. Forward and Inverse Position Kinematics

To determine task space configuration with joint configuration given, we derived a forward kinematics solution to convert joint values to end-effector position. Here we first labeled the coordinate system of each joints of the robot arm using DH convention, as shown in Fig. 1, and derived DH parameters for calculating forward kinematics, as shown in Table I.

After deriving DH parameters, we formed transformation matrices A_1, A_2, and A_3 and the corresponding transformation matrices with respect to the base frame T_1, T_2, and T_3. Given that these homogeneous transformation matrices are functions of θ_1, θ_2, and θ_3, once we acquire the joint values, we could calculate the position for the end-effector.

To derive a joint space solution to reach a specific point in task space, we derived two inverse kinematics solutions. The first one makes use of geometric approach while the second method employ numerical methods.

Thank you to Professor Farzan and SAs that help us throughout this project.

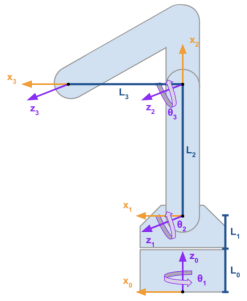

Fig. 1. Coordinate Systems labeled by DH Convention

TABLE I
Table of DH Parameters

DH Parameters				
i	θ	d	a	α
1	θ_1^*	$L_0 + L_1$	0	- π/2
2	θ_2^* - π/2	0	L_2	0
3	θ_3^* + π/2	0	L_3	0

For the geometric approach, as shown in Fig. 2 attached in the Appendix, the position of the task space position is given as $O_w = [X_w, Y_w, Z_w]$. Then we could calculated $\theta_1 = \arctan(Y_w/X_w)$, making use of the projection onto the $x - y$ plane. Next, we could derive $\cos(\theta_2) = (r^2 + s^2 - L_2^2 - L_3^3)/(2L_2L_3)$ by using law of cosines. While $\sin(\theta_2) = \sqrt{1 - \cos(\theta_2)^2}$, we got the value of $\theta_2 = \arctan(\sin(\theta_2)/\cos(\theta_2))$. Finally, to get the value of the last joint θ_3, we made use of the law of cosines and trigonometry to find $\theta_3 = \arctan(s/r) - \theta$, where $\theta = \arctan(L_3 \sin(\theta_3)/(L_2 + L_3 \cos(\theta_3)))$.

For the numerical method approach, we first made a guess of a joint configuration that might give the desired end-effector position, and then we used forward kinematics to compute the end-effector position of the guess configuration. Next, we compared the current task space position with the desired position. If the difference is not small enough, we move the current position toward the desired position and iterate the process until the difference is smaller than tolerance.

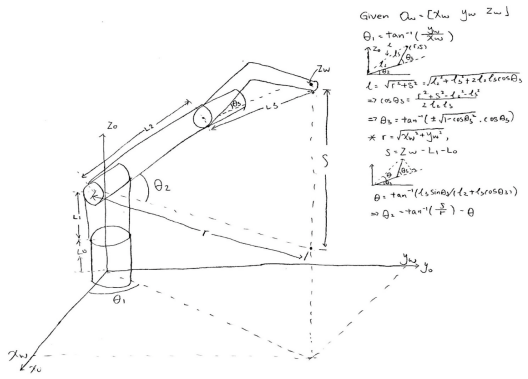

Fig. 2. Inverse Kinematics Derivation

B. Forward and Inverse Velocity Kinematics

In the previous section, we derived a relationship between joint space configuration and task space position. Here we calculate Jacobian matrix of the robot arm to relate joint velocities and end-effector velocities.

By geometric approach, we formed the Jacobian matrix, as shown in Table II. Then we used forward differential kinematics to calculate end-effector motion with given joint configuration and velocities.

To calculate the joint velocities based on end-effector motion, we used the inverse Jacobian matrix. Here we noticed that singular configuration is a problen when doing inverse differential kinematics. Therefore, we programmed the robot to constantly make aware of the determinant of the Jacobian matrix to avoid running into a singularity.

TABLE II
TABLE OF JACOBIAN MATRIX

DH Parameters			
	Joint 1	Joint 2	Joint 3
J_v	$Z_0 \times (O_n - O_0)$	$Z_1 \times (O_n - O_1)$	$Z_2 \times (O_n - O_2)$
J_w	Z_0	Z_1	Z_2

C. System Architecture

The robot arm that we used is a serial 3 DoF spherical manipulator with three servo motors to control revolute joints and another servo on the gripper to pick up objects. To obtain the joint angles and velocities, the robot would need to read the motor encoder values and perform calculation using a 32-bit ARM Cortex-M4 microprocessor (Microchip ATSAMD51). However, in this course, we had been provided with pre-compiled embedded firmware with basic functions so that we could focus on the high-level logic of the robot manipulation.

Another major component of the pick and place system is the camera. We used MATLAB add-in toolbox to calibrate and communicate with the camera so that we could incorprate vision-based manipulation in our project.

D. Robot Arm Stick Model

In order to visualize the system, we created a 3D stick model to mimic the robot arm where each link is represented by a line and each joint and the end-effector is represented by a point. In addition, we labeled the axes of the coordinates of each frame, including base, end-effector, and intermediate frames, by vectors. In addition, we also included the end-effector velocity as a vector to the stick model so that it could reflect in which way and what speed the end-effector is moving.

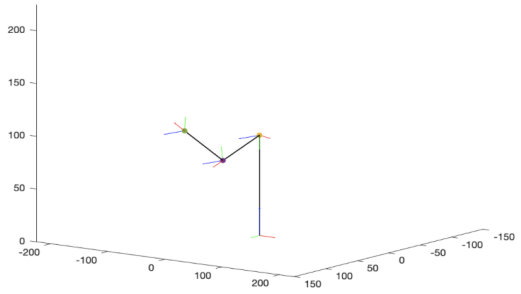

Fig. 3. Robot Stick Model of joint configuration $[15, 250, -60]$

E. Image Processing Pipeline

To be able to recognize objects and pick them up, we needed to develop an image processing pipeline. When designing this pipeline, we prioritized simplicity to aid in debugging and increase processing speed. A number of different HSV masks were created with MATLAB's Color Thresholder app to filter out balls and dry-erase marker caps of varying colors. One of these masks would be used to filter out pixels that do not encompass the desired object. Then, this filtered image would pass through the MATLAB vision toolbox's Blob Analysis, where centroids of blobs would be recognized. Although the input image often contained noise, this noise would not be recognized as a blob in the Blob Analysis, so it was not an issue.

III. RESULTS

IV. DISCUSSION

A. Intrinsic Camera Calibration

To perform camera calibration, we would need to calculate extrinsic parameters such as homogeneous transformation H_{camera}^{world}, intrinsic parameters such as focal length, the optical center (principal point), skew coefficient, and distortion coefficients for radial and tangential distortions.

In our project, we used the MATLAB Single Camera Calibrator App to calculate intrinsic camera matrix. First, we took forty pictures of the checker board using the camera and checked if the coordinates in all the pictures were aligned. Then we evaluated the calibration by examining extrinsic parameters, mean reprojection error, and undistorted

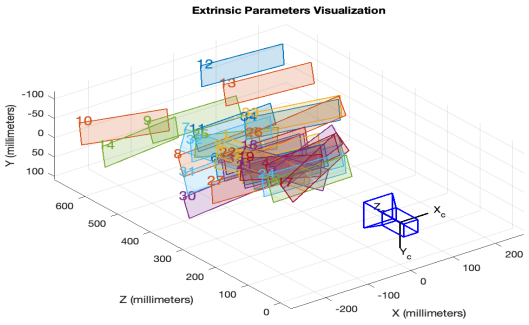

Fig. 4. Extrinsic Parameters Visualization

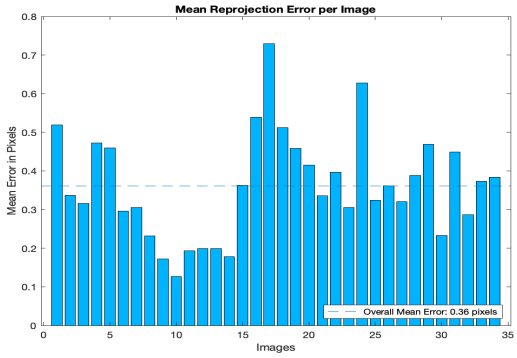

Fig. 5. Mean Reprojection Error per Image

image. The extrinsic parameters are in fact the frame of the checkerboard in the pictures. Mean reprojection errors are the distances between detected and associated reprojected points. The calibration is considered satisfactory when a mean reprojection errors of less than one pixel is achieved. Here, as shown in Fig. 3, we achieved a mean value of 0.4 pixels, which showed that the camera was well calibrated. Finally, we exported camera parameters and corresponding MATLAB script so that we could easily calibrate the camera every time we would like to operate the robot arm.

The rationale behind the checkerboard calibration method, in our opinion, is that there is a relationship between the pixel coordinates $[x, y]$ and the camera coordinates $[X_c, Y_c, Z_c]$, and let us denote this relationship as a function $g(f, c, s)$. Here the parameters of the function are all intrinsic parameters of the camera, where f is focal length, c is optical center, and s is skew coefficient. Once we got sufficient number of pixel coordinates and camera coordinates by taking at least twenty pictures, we could derive the relationship between the two quantities and calculate the intrinsic parameters.

B. Camera - Robot Registration

Once we calculated the intrinsic parameters of the camera, we performed extrinsic calibration to establish camera-robot registration. We first obtained the extrinsic calibration parameters in an transformation matrix and a position in the

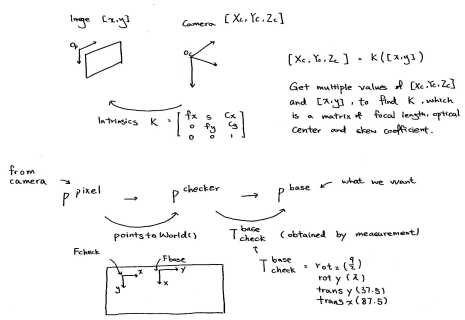

Fig. 6. Intrinsic Calibration

pixel frame (P^{pixel}). Then we used MATLAB built-in function to compute the position in the local reference frame of the checkerboard ($P^{checker}$). Finally, we measured the distance between the robot and the camera to derive a homogeneous transformation matrix of the checkerboard frame with respect to robot's base reference frame ($T^{base}_{checker}$). Once we multiplied to position in the checkerboard frame by the transformation matrix , we would obtain the position to the robot (P^{base}).

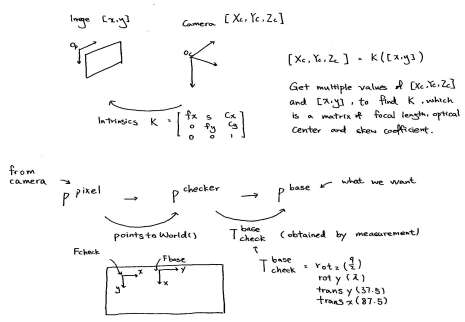

Fig. 7. Extrinsic Calibration

One thing here we noticed was that, in practice, this method would only work well for z = origin of the checkerboard reference frame. This was because if z axis is above the checkerboard, it is hard to obtain the transformation matrix of the checkerboard frame with respect to robot's base reference frame ($T^{base}_{checker}$) as it was more difficult to measure the distances between these two frames.

C. Object Detection and Localization

In this section, we performed image processing and derived the centroid location of the balls. We first un-distorted the frames from the camera using the calibration parameters. Then we enhanced the image by image segmentation. Once the image is processed, we could dtermine the centroid position of the balls by using the MATLAB built-in function.

To detect and localize balls of different colors, we created a mask for each color using MATLAB built-in functionality.

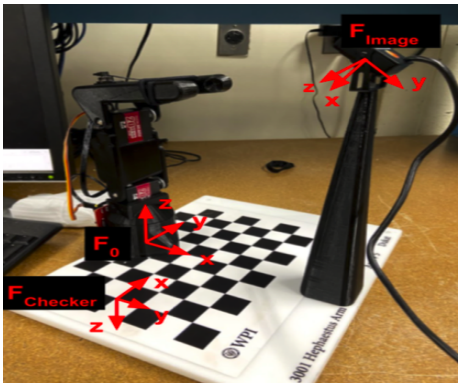

Fig. 8. Coordinate frames of the Robot-Camera System

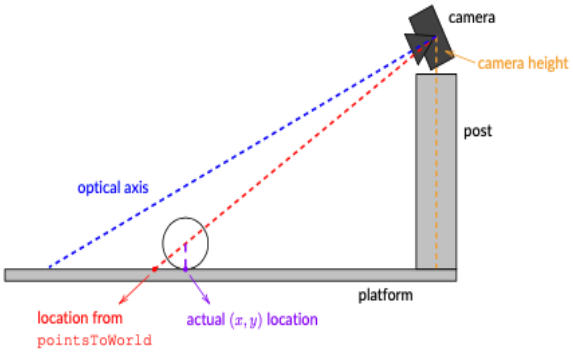

Fig. 9. Side view of the Robot-Camera System

Here we used HSV color space instead of the widely-used RGB color space. The reason was that the lighting condition in the lab do not remain the same. Sometimes when we switch our spot, we found the color of the balls different in the camera view due to the lighting. As demonstrated in class, by using RGB color space, the color thresholds are hard to obtain as slight change in lighting will result in large variance in the values in all three RGB channels. To prevent this issue, we chose to HSV color space it is easier to threshold for a color using Hue channel, and the lighting will only affect the value (or brightness) channel.

During the experiment, we found out that black balls cannot be used for the pick and place system. This is because that when we create a mask for a black ball, it will be easily misinterpreted by the black blocks on the checkerboard. Despite that we could tune the HSV parameters for the mask, the camera would still easily mistake the black squares as black balls. Therefore, to prevent the autonomous robot pick and place system grabbing imaginal black balls non-stop, we did not use black balls for the purpose of this project.

D. Object Localization

Obtaining the position of the centroid of the balls, we took into consideration of the height of the balls. Through measurement, we found the radius (or height for pick up location) is 10mm. Then we used our knowledge of linear algebra to compute the actual (x, y) position based on the location from pointsToWorld().

Here let us denote the location from pointsToWorld() as $P^{checker}$. We first defined a new coordinate system called F_{post} whose origin is at the central intersection of the post and platform and axis orientations are the same as the checker frame ($F_{checker}$). Through measurement, we found the transformation matrix between the two frames mentioned above ($T_{post}^{checker}$) and the camera position with respect to the post frame (C^{post}). With homogeneous transformation, we could derive the position for the point (P^{post}). Now that we obtained the position of both the point and the camera with respect to the post, we could set up a vector from the point to the camera ($\vec{PC}^{post}$). Since both the height of the ball and the camera is known, we could use similar triangle theory to calculate the actual location of the ball with respect to the post ($P^{post'}$). Finally, through measurement, we could get the transformation matrix from the robot base frame to the post frame and we could use matrix multiplication to get the actual location with respect to the frame of the robot arm.

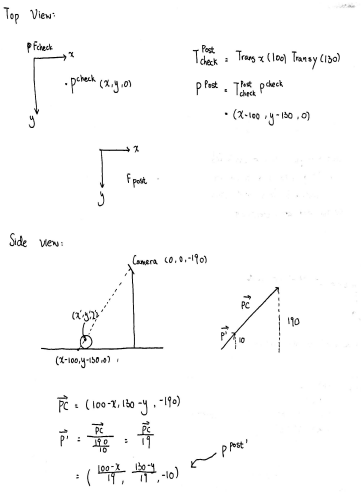

Fig. 10. Compensation For Height

E. Final Project Challenge

For the final project we implemented the fully automated pick and place robot sorting system that could constantly detect and localize a ball, move to a position through a quintic trajectory to pick up the ball, and move a place based on its color to drop it off.

To accomplish the challenge, we implemented a state machine, which is extremely helpful to complete a logistic task and for debugging. First, the robot would enter a "looking" state where the robot move to the zero position ($[0, 0, 0]$ in joint space); it would use the camera to detect whether there is a ball on the checkerboard; if not, it would keep looking; otherwise, it would set the state to "waiting for placement". Next, in the

"waiting for placement" state, it would try to localize the ball twice, as long as the 2 locations of the ball are within tolerance (1mm), meaning that it was not moving, it would calculate the position of the ball with respect to the robot frame, generate a quintic trajectory in task space to move to a point above the ball, and move to the "vert traj" state. However, if the 2 locations are not similar enough, which means that the ball is moved, it would return to "looking" state to keep detecting for a ball. Once the robot is in the "vert traj" state, the robot would move to the point above the ball following the trajectory generated in the previous state. Having it move to the point above instead of moving directly to the pick up position would allow the robot to avoid obstacles. Upon reaching the position, the robot will open the gribber and set the state to "grab traj". In the "grab traj" state, the robot closes the gripper and move to "grabbing" state in which the robot generate a trajectory to move to a position where it drop the ball categorized by its color. Finally, it will enter the "place traj" state where the robot move to the drop off position and proceed to "placing" state that it opens the gripper and drops the ball.

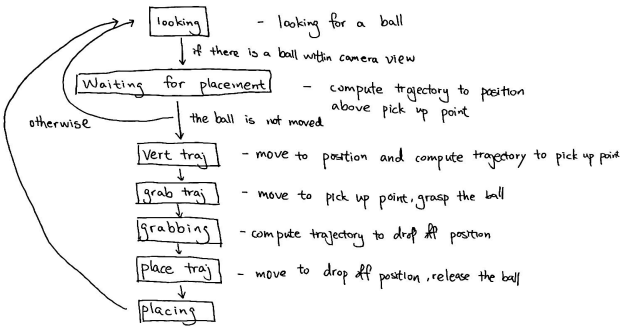

Fig. 11. State Machine Diagram

Here we used quintic trajectories in task space. Using quintic trajectories allows the robot to follow a trajectory with zero starting and ending velocity and acceleration so that the motion is smooth around via-points with no jerk. The reason we were using task space trajectory is that it follows a straight time when we perform motion planning on the end-effector. This would benefit the robot by saving time and avoid hitting obstacles. For the first extra credit, although we did not have time to complete, we made a mind experiment. In our opinion, we could use the velocity trajectory to accomplish dynamic camera tracking. The input to the tracking function would be the position of the ball in the camera view, which is a variable, so that the robot arm would keep approaching the ball with a certain speed until it reaches the ball (and the ball stop moving). After it reached the ball, it would did the same task to grasp, move, and drop.

As an extra credit, we also programmed the robot so that it could also pick up blue markers. The approach were very the same as picking up the balls. We first constructed a mask for a blue marker, then we added the functionality to our main code, and it would behave the same as it were picking up a ball.

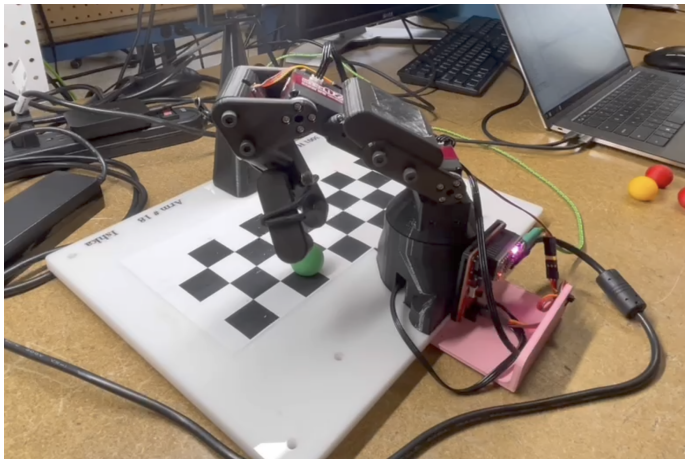

Fig. 12. Robot Arm picking up a green ball

CONCLUSION

In this project, we developed an autonomous pick and place system for sorting balls of different colors. We implemented knowledge from lectures such as position kinematics, velocity kinematics, motion planning, and computer vision into practice. In addition, we had a hands-on experience programming a robot arm which be beneficial for our future career.

Here we want to appreciate Professor Farzan and SAs for their help and guidance throughout the project. They have tackled the low-level embedded communication and developed basic function so that we could focus on the high-level logic.

In addition, we would like to propose some suggestions for future project:

- Explain in more details about what should be included in results and discussion. Sometimes it could be confusing that students just combine these two sections.
- To derive position kinematics, one task in the lab could also be picking up a ball in a fixed position without a camera so that students could get a heads up the algorithm of ball picking and this could also verify the position kinematics solution as well.
- Discuss a bit about the system architecture of the robot arm. Although we are told to treat the system as a black box, we are interested in knowing more about the robor. Sometimes when the robot run into a technical error, we have no clue how to solve it on our own. Thus, it could be beneficial to spend sometime in the very first lab to talk about the system architecture.

TABLE III
CONTRIBUTION

	Calvin Page	Puen Xu	Arturo Lemos Lee
Contibution			

CONTRIBUTION

APPENDIX

Figures

Project Video

Git Repository