18%/20% Teach and Repeat: -2% no plot of low and high velocity; -4% no workspace plot of smoothing; +2% +++ considered quintics; +1% +++ reported success stats.
25%/25% Grasp execution: excellent! did all required parts well
27%/25% Camera calibration & Blob detection: excellent! did all required parts well +2% +++ orientation
28%/30% Autonomy: -2% issues with causality in discussion
98% TOTAL >>NEARLY PERFECT SCORE<<

***Abstract*—We propose a methodology of automating an articulated arm with 5 degrees of freedom in a constrained workspace, to pick up and place blocks from the workspace in specific configurations. We discuss the design of the gripper for picking and placing blocks, and also discuss corresponding forward and inverse kinematic algorithms. For smoother kinetics we constrain the joint motion to a spline function and compare the results through teach and repeat tasks. For perception of the workspace and block detection, we use a Kinect based vision system. Finally, with the help of predefined competition tasks, we further deliberate the execution of complex tasks using motion planning algorithms and automate the the tasks as per the events of the competition.**

Keywords- Robotic arm, Inverse Kinematics, Quadratic Discriminant Analysis

I. Introduction

ARTICULATED robotic arms have been increasingly integrated into operations, freeing human laborers from highly repetitive work. To that end, we build a 4 DOF articulated robotic arm with 1 extra DOF on the end effector. The articulated robot can detect blocks of different colors on a fixed plane, and finish the tasks of stacking, unstacking, and lining the blocks. The arm is built using 3 MX-28[1] servo motor (to drive base, shoulder and elbow), 1 AX-12[2] servo motor (to drive the wrist) and 2 XL-320[3] servo motors (to drive the gripper). Blocks are perceived using RGBD image detection by a Kinect camera mounted above the workspace.

We first present the methodology used in Section II and then present results for each component of the robot in Section III, and conclude with discussion in Section IV.

II. Methodology

Here we present how the different subsystems of the robotic arm were built: motion planning in Section II-A, grasping in Section II-B, object perception in Section II-C, and autonomous execution in Section II-D.

A. Teleoperation & Motion Planning

In this section we describe the implementations used to control the arm via teleoperation and autonomously.

[1]The motor can be found athttp://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28at_ar.htm

[2]The motor can be found athttp://support.robotis.com/en/product/actuator/dynamixel/ax_series/dxl_ax_actuator.htm

[3]The motor can be found athttp://support.robotis.com/en/product/actuator/dynamixel_x/xl_series/xl-320.htm

1) Teleoperation: We first implemented controls in the graphical user interface (GUI) provided in the ROB 550 Arm Lab [1] to send direct commands for the servos to set the arm's joint angles. For each joint in the arm we used one slider to set the current angle, and limited the allowed range of angles to the physical limits measured in Table I. A separate pair of sliders was used to set the servo torque and speed.

Joints	Min θ	Max θ
Base	-180	179.9
Shoulder	-118	118
Elbow	-115	120
Wrist	-120	120

TABLE I: Minimum and maximum angles in degrees for individual joints.

2) Splines: To autonomously move the arm to various locations, we implemented several methods and sent direct commands to the servos as the baseline. The velocities and accelerations of joints executing direct commands are not known, so splines present an alternative method of motion planning wherein these quantities are given by the fitted spline coefficients. We consider splines parameterized by time and mapping to a single joint angle. The entire arm undergoing spline-based motion planning is thus given by a fitted spline for each joint. We consider splines of form:

$$q(t) = \sum_{i=0}^{5} c_i * t^i \tag{1}$$

We first consider cubic splines, where $c_4 = c_5 = 0$ in (1). This leaves four coefficients for which to solve, which we specify with an initial and final position, and initial and final velocities set to 0. Solving this system of equations yields the coefficients for a function that is four times differentiable. When following a path with multiple waypoints, the splines may be 'tied' together (at 'knots') such that the ending position of one spline is the starting position of the next. Taken together, the piecewise cubic function is differentiable only once since the velocities are now only guaranteed to be continuous at the knots. This spline is shown following a simulated path in Figure 1(*red*).

With quintic splines we additionally specify initial and final acceleration to be 0, leave c_4, c_5 unconstrained, and the piecewise quintic function is twice differentiable Figure 1(*bottom, magenta*).

Lastly we consider cubic splines, but instead of specifying velocity at each internal knot, the final velocity of each spline is set equal to the initial velocity of the next and similarly for accelerations. The unspecified velocities allow for more natural looking motion, removing the stops made at intermediate waypoints. The boundary conditions preclude the solving of each spline independently of the others so we implemented the variable elimination algorithm given in Appendix A of [2] with *cubic_spline.py* in the submitted code, lines 10-107.

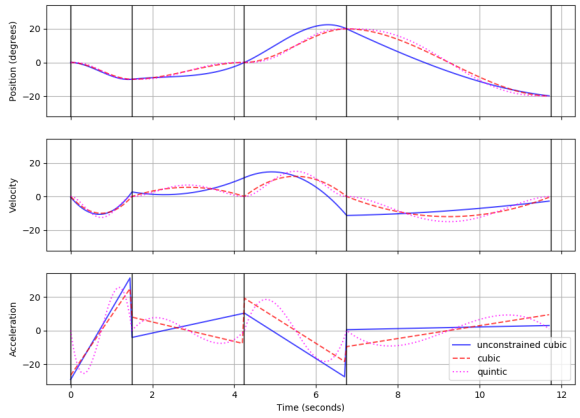


Fig. 1: Simulated spline following for joint angle waypoints [-10, 0, 20, -20] starting from 0 (in degrees). The unconstrained cubic spline is not constrained to have 0 velocity at waypoints, while the quintic spline being higher order maintains continuous acceleration.

3) Obstacle Avoidance: Splines enable smooth motion between points, but the provided constraints do not incorporate obstacles. High level motion planners are used for this purpose, to find a path to a given destination point such that each segment in the path is constrained to be free of obstacles. Each segment is then traversed with a local planner, such as a spline. We implemented Probabilistic Roadmap (PRM) (Ch.7 [3]), with a sample path shown in Figure 2. We found that while even for a 5 dimensional arm, uniform sampling across the configuration space was prohibitively expensive computationally, causing the arm to pause for more than several seconds. Though this could be reduced with more effective sampling, we did not use the method in the challenges due to time constraints.

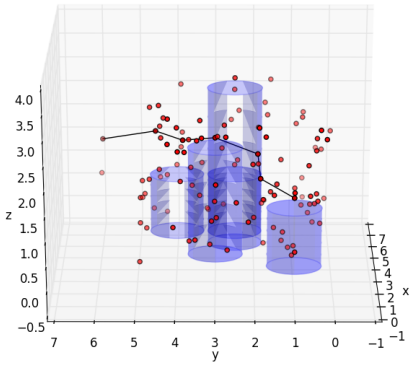


Fig. 2: The path (*black*) through nodes (*red*) found by Probabilistic Roadmap avoids obstacles (*blue*).

B. Grasp Execution

1) Gripper Design: The gripper is the end-effector attached to the end of last linkage on the robotic arms. The purpose the gripper is to pick up the blocks on the working plane and place them to finish the lining and stacking tasks. We use 2 XL-320 motor to drive the gripper with stall torque of 0.39 N·m[4]. Limited stall torque suggests that the components should be light and friction among joints should be small. Therefore we designed the gripper to have a two clamps structure connected by 14-teeth gears as shown in Figure 3.

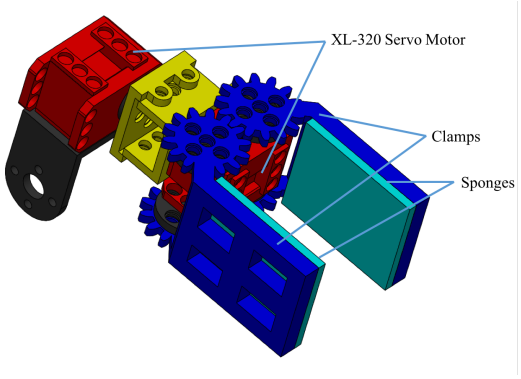


Fig. 3: Gripper design

2) Forward Kinematics: Forward kinematics is the mapping of joint angles coordinate system $\boldsymbol{\theta} = \{q_1, q_2, q_3, q_4\}$ to the workspace coordinate system, $\mathbf{X} = \{x, y, z\}$. Forward kinematics is essential in path planning, especially to impose restrictions on $\boldsymbol{\theta}$ to avoid collision in workspace measured in $\mathbf{X}$. We follow the Denavit-Hartenberg (DH) convention to select frames of reference. The coordinate frames are located on rexarm

[4]Refer footnote 3

as shown in Figure 4. We followed DH conventions to form Table II using four parameters θ_i, d_i, a_i and α_i that are associated with link i and joint i.

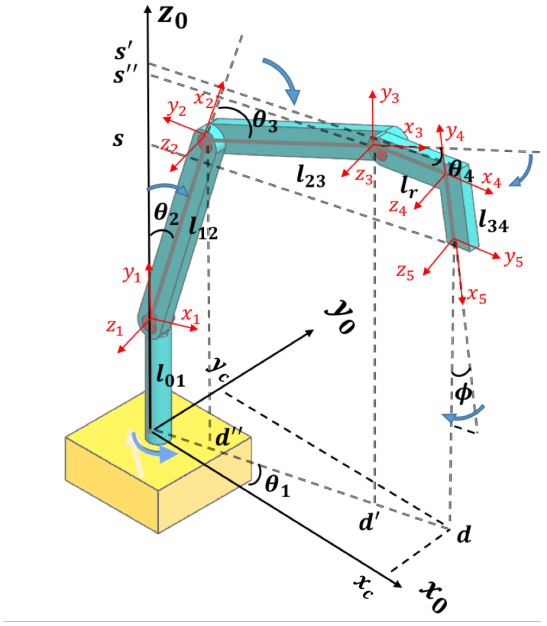


Fig. 4: Configuration for Forward and Inverse Kinematics. The blue arrow is negative direction in which joint angle θ_i changes. $X_0Y_0Z_0$ is the base frames.$X_iY_iZ_i$ $(i = 1, 2, 3, 4, 5)$ are frames associated with each link. joint angle θ_i is measured as the angle between axis x_{i-1} and x_i following the right hand rule.

Joint Number	θ_i	$d_i(mm)$	$a_i(mm)$	α_i
1	$\theta 1$	l_{01}	0	$\pi/2$
2	$-\theta_2 + \pi/2$	0	l_{12}	0
3	$-\theta_3$	0	l_{23}	0
4	$-\theta_3$	0	l_r	0
5	$-\pi/2$	0	l_{34}	0

TABLE II: Denavit–Hartenberg parameters specifying the forward kinematics of the articulated arm. Link lengths have values: $l_{01} = 115$, $l_{12} = 97.63$, $l_{23} = 98.54$, $l_r = 42$ and $l_{34} = 98.99$

In this convention, each homogeneous transformation A_i is represented as a product of four basic transformations given in equation 3.10 from [3]. We use these homogeneous transformations to form transformation matrix linking end effector coordinates with base frame coordinates given by:

$$T_0^5 = A_0A_1A_2A_3A_4 \tag{2}$$

3) Inverse Kinematics: Inverse Kinematics is used to map workspace coordinate system $\mathbf{X} = \{x, y, z\}$ to joint angles coordinate system $\boldsymbol{\theta} = \{q_1, q_2, q_3, q_4\}$. For the rexarm this can be done geometrically if the orientation of end effector ϕ as shown in Figure 4 is known. The solution of $\boldsymbol{\theta}$ is not unique given $\{\mathbf{X}, \phi\}$ and has multiple solutions and will be evident in the following geomtric analysis where we map $\mathbf{X}$ to $\boldsymbol{\theta}$.

$$\theta_1 = atan2(x_c, y_c) \tag{3}$$
$$d = \sqrt{x_c^2 + y_c^2} \tag{4}$$

As seen in Figure 4 we can calculate s' and d' given pose ϕ.

$$s' = s + l_{34}cos(\phi) \tag{5}$$
$$d' = d - l_{34}sin(\phi) \tag{6}$$
$$s'' = s + l_{34}cos(\phi) - l_r sin(\phi) \tag{7}$$
$$d'' = d - l_{34}sin(\phi) - l_r cos(\phi) \tag{8}$$
$$cos(\theta_3) = \frac{d''^2 + (s'' - l_{01})^2 - l_1^2 - l_2^2}{2l_1l_2} := D \tag{9}$$
$$\theta_3 = atan2(\pm\sqrt{1 - D^2}, D) \tag{10}$$

Equation 10 shows that θ_3 has two solutions highlighting that the solution of $\boldsymbol{\theta}$ is not unique.

$$\theta_2 = atan2(d'', s - l_{01}) - atan2(l_{23}sin(\theta_3), l_{12} + l_{23}cos(\theta_3)) \tag{11}$$

$$\theta_4 = \pi/2 - \phi - \theta_3 - \theta_2 \tag{12}$$

4) Dextrous Workspace: To check for reachable points in the workspace, three gripper configurations, $\phi = 0$, $\phi = \pi/2$ and $\phi \in (0, \pi/2)$ are used. The first two configurations are preferred because block edges in these conditions are parallel to the clamps' edges making the block pick up and drop tasks accurate. The $\phi \in (0, \pi/2)$ is only used at the positions where first two gripper configuration cannot reach. To illustrate the working range of each configuration, the joint angles θ_1 to θ_4 are varied according to their limitations from $-2\pi/3$ to $2\pi/3$. The working space for $\phi = 0$ rad and $\phi = \pi/2$ rad configuration is shown in the Appendix B Figure 6.

C. Object Perception

Using an overhead mounted Kinect sensor setup [1] providing RGB and depth images, we implemented a block detection system in order to sense blocks in the workspace. The information for each detected block consists of the position of the block in world coordinates, color, orientation, and the position of the block if it is in a stack (stack level), each of which is used in autonomous execution (Section II-D). We detail this system in three parts. In Section II-C1 we detail the camera calibration used to find object coordinates in the world frame given the corresponding coordinates in the RGB image, in

Section II-C2 we detail how blocks were detected, and in Section II-C3 how the color for each block was determined.

1) Camera Calibration: Determining world coordinates from positions in the RGB image was accomplished in three steps. We first assume that the projection of the workspace to the image plane can be approximated by an affine transform (Ch.2, [4]). The image plane is mounted parallel to that of the workspace, resulting in the projection being "close" to orthographic and thus the points in the parallel planes are related by a transform "close" to affine.

Due to an offset and difference in scale between the RGB and depth images, an affine transform was used for registering the two. The xy-plane world coordinates of pixels in the RGB image were obtained by registering the RGB image to the workspace via another affine transform. The matrices corresponding to affine transforms are invertible, so we may freely convert between world, RGB, and depth coordinates. Henceforth, in algorithms we assume that we can perform any necessary conversions between coordinates.

We then find the height of objects in the workspace by measuring depth to the board surface. We convert the 11-bit depth values from the depth image into distance in millimeters with the formula:

$$distance(depth) = 123.6 + 1000 * tan(depth/28425 + 1.1863) \quad (13)$$

Since the workspace board represents the xy-plane in the world frame, we determined world frame height with:

$$z = distance(point) - distance(board)$$

We observed that blocks measure 38+-1mm to each side and that as these are the only objects of in the workspace other than the arm itself, heights of all objects may be described in terms of stack level. Detected heights of single blocks were comparatively noisy and ranged from 4mm to 6mm, so we thresholded the detected height into bins for each stack level. Using 6mm as block height, heights within 3mm of a given stack height are assigned as such.

2) Block Detection: We observe that the arm workspace is a plane that is parallel to the Kinect sensor's image plane, allowing for blocks to be separated from the workspace board at a depth threshold. Motivated by this, we detect contours in the thresholded depth image to detect the blocks. A minimum area rectangle is fit to each of the contours to obtain block orientation, and the mean of these points is taken as the block position. In order to filter out false positives, we find the range in which block areas may lie $A = [area_{min}, area_{max}]$, and the rectangular region occupied by the arm at rest with a fully open gripper $R = R_X \times R_Y = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$. These are used because links on the arm may be mistaken as blocks by the algorithm. Blocks outside A or within R are filtered out to obtain the final set of block detections. The algorithm pseudocode is shown in Algorithm 1.

Algorithm 1 Block Detection

Require: τ, A, R
  $D =$ threshold depth image by τ
  Detect contours in D
  Fit minimum area rectangles fit to contours
  Remove if detection area $\notin A$
  Remove if detection center $\in R$

3) Color Classification: We being by building a generative model for classifying the blocks by color. Assume for each color c, each pixel x is identical and independently distributed (*iid*) according to: $x \sim N(\mu_c, \Sigma_c)$. This model is also known as Quadratic Discriminant Analysis (QDA). We also assume that each color is *a-priori* equally likely, since blocks of any of the colors may be placed in the workspace. Each detected block has a number of pixels, $x_1, ..., x_n$, thus we are aiming to find:

$$c^* = \arg\max_c \; p(y = c|x_1, \ldots, x_n) \quad (14)$$

By Bayes rule,

$$p(y = c|x_1, \ldots, x_n) = \frac{p(x_1, \ldots, x_n|c)p(c)}{Z} \propto p(x_1, \ldots, x_n|c) \quad (15)$$

where the last line follows from the fact that $\frac{p(c)}{Z}$ is a constant due to our assumption of a uniform prior. Since each pixel is *iid*, $p(x_1, \ldots, x_n|c) = \prod_{n=1}^{n} p(x_i|c)$. The classification for the block then becomes:

$$c^* = \arg\max_c \prod_{n=1}^{n} p(x_i|c) = \arg\max_c \sum_{n=1}^{n} log(p(x_i|c)) \quad (16)$$

The last equality is important because calculating the log-likelihood allows us to avoid numerical underflow issues. These steps form Algorithm 2.

Algorithm 2 Color Classification

Require: μ_c, Σ_c for $c \in Colors$, pixels $x_1, \ldots, x_n$
  for $c \in C$ **do**
    Calculate $\sum_{n=1}^{n} log(p(x_i|c))$
  $c^* = \arg\max_c \sum_{n=1}^{n} log(p(x_i|c))$

D. Autonomous Execution

Since no obstacle avoiding path planner was used, we avoided collisions by moving vertically to specified heights before moving to final xy-plane locations, and again moving vertically to the final position. The 'safe heights' were calculated as the height of the tallest stack of blocks plus a set offset. Blocks can then be picked and placed without collisions provided accurate detection of the number of blocks in each stack. In order to detect blocks, we first return the arm to the initial with the clamp fully open before capturing an image for detection. The strategy used for each task is written in Algorithm 3.

Algorithm 3 Generic Task Strategy

while not done **do**
  Capture image at initial position *if needed*
  Find block of interest
  pick_and_place_block(
   initial location, safe height, final location)

Tasks 3 and 4 require first finding an area in which to build a block structure, and destacking blocks to find the colors needed to build the structures in color order. We approached this by detecting open regions of space on the workspace board, clearing a designated area of any blocks, and then destacking blocks until all colors were visible. Any blocks to be moved in these stages were placed in open areas. At this point tasks 3 and 4 follow the same strategy as the others.

III. Results

We now present evaluations for each subsystem in turn: motion planning in Section III-A, grasping in Section III-B, object perception in Section III-C, and autonomous execution in Section III-D.

A. Teleoperation & Motion Planning

We evaluated the ability of the cubic and quintic splines to plan smoothly by testing their performance on the game "Operation". The goal of the game is to insert the end effector into a series of 11 slots in the game board without touching the edges of the slots. A series of waypoints was recorded through kinesthetic teaching and the path was replayed for each method. The time length specified for moving from one waypoint to the next was determined by dividing the maximum angle change between the waypoints by a set angular speed, so a higher speed would result in shorter times. Each method was tested on "Operation" once at a slow rate of 20 degrees/second and once at a faster rate of 60 degrees/second. The cubic and quintic splines performed similarly and each outperformed direct commands, shown in Table III. Due to time constraints, the final cubic spline method was successfully implemented on the arm, and the other spline methods were not used in attempting the competition tasks.

	Direct Commands	Cubic Spline	Quintic Spline
Slow	8/11	11/11	11/11
Fast	6/11	10/11	10/11

TABLE III: Motion planning method performance on "Operation". The cubic and quintic splines perform equally well, and outperform direct commands to the servos at both speeds.

B. Grasp Execution

The inverse kinematics discussed in section II-B3 were used to determine the ideal (from competition's perspective) range of workspace coordinates in which we can position our arm accurately. We found the ranges of distance "d" from the base in XY-plane in workspace, and height "z" along the Z-axis, to be $d \in [150, 275]$ and $z \in [0, 200]$. To compute the maximum distance and height, the orientation of the gripper that could pick up 10 out of 10 blocks successfully is selected. Table IV shows the results of the orientation we used for the range of radius and heights in the workspace.

d (mm)	z (mm)	ϕ (radians)
[150,225]	[0,100]	0
[150,275]	(100,150]	$\pi/2$
[225,275]	[0,100]	$\pi/2$

TABLE IV: Chosen orientations for given radius and height of the end effector position in workspace coordinates. "d" is the distance from base in XY-plane, "z" is the height measured along Z-axis and "ϕ" is the orientation of the gripper as mentioned in 4

C. Object Perception

Here we describe how we determine the correspondences used for the affine transforms, set parameters for Algorithm 1, and the dataset used for fitting the color classifier.

1) Performing Camera Calibration: We collected 9 points to estimate the affine transforms; each corner of the workspace board was collected in the RGB image coordinates and depth image coordinates, and a single point on the surface of workspace board in the depth image was collected. The corners of workspace board in world coordinates were also measured, and the affine transforms were estimated using the corresponding corner points. The depth value at the workspace board point was used as the workspace board's depth.

2) Finding Block Detection Parameters: Using the provided depth images in the ROB550 Arm Lab folder as input, we set parameter values such that all of the contained blocks were detected. We found that $\tau = 710$, $A = [500, 1400]$ were one such setting, and used this for our experiments. The arm at open-grip resting occupied $R = [-50, 50] \times [-70, 70]$, in millimeters.

3) Evaluation of Detections: We evaluated the block detection system presented in Section II-C2 by comparing the detected block centers to the real centers in world coordinates. Blocks were placed every 150mm in a grid across one quadrant of the workspace, and different heights were used to evaluate the detected height as well (see Appendix D). The 3D distance to actual location was 61.5+-25 mm. Since all stack levels were correctly detected we also report the distance along the world coordinate xy-plane, found to be 20+-6 mm. To see to what extent camera calibration affects the block detection system and to evaluate the calibration itself, we manually selected centers of the blocks in the GUI to obtain world coordinates according to the calibration and compared these to the actual coordinates. The error in 3D distance was 60.5+-26 mm while the error in the pl was 18+-5 mm. We see that the majority of error in ing world coordinates from images stems from the calibration. Additionally, accurate estimates of stack level reduce the error significantly.

4) Color Classification: The color classifier in Section II-C3 was evaluated with a dataset of 69 images, containing 554 visible blocks and collected using the mounted Kinect. This was separated into a training set of 63 images (507 blocks) and a test set of 6 images (47 blocks). The underlying QDA model was fitted with the implementation from [5], and achieved a per-pixel classification accuracy of 78.07%. The full model, however, correctly clas l 97.83% of the training set blocks (496/507), and all of the test set blocks correctly.

D. Autonomous Execution

We evaluate the autonomous performance of the arm with scoring according to the ROB 550 Arm Lab competition [1]. This is because the competition rules award points on the basis of both speed and accuracy, making it a natural metric with which to judge the arm. For Task 1 & 2 we were awarded 90 and 92.5 points respectively. We were awarded extra points for task 2 because we completed the task in less than 40 seconds.

In Tasks 3 & 4 we received no points due to failure in unstacking the blocks. We observed that we needed one extra state to change the orientation of the gripper when the arm is in neutral upright position. Without this state the gripper would fail to change its orientation to align with the stacked blocks and instead topple them.

We were awarded 200 points for Task 5 for building a four level pyramid in 180 seconds, but our arm was unable to build a larger pyramid due to its occasional dropping of blocks.

IV. Discussion

In this project we designed a gripper for picking up and placing blocks. We implemented forward and inverse kinematics accurately to move the blocks as it was demonstrated by the pick and place events of the competition. Overall, we were able to autonomously able to complete the tasks given in the competition. The largest difficulties stemmed from inaccurate resolution of block coordinates in world space and the lack of state machine logic to fully control gripper orientation, which caused Tasks 3 and 4 to be infeasible. The results from Section III-C3 show that the error in perceived block coordinates is largely due to the camera calibration, with the block detection adding on average only 3 mm of error over that of manually specifying the blocks in image coordinates. Since the blocks average 38 mm on each side, the 20 mm of error attributed to calibration suggests that the gripper performed very well under these adverse conditions. The approximation by an affine transform made in Section II-C1 in findi rld coordinates could be improved possibly by cor the radial distortion seen towards the edges of the images (see Appendix C Figure 7a).

One success was the color classification of the detected blocks. Despite the applications of matte paint, the blocks would reflect the ceiling lights, causing glare resulting in washed-out colors when captured by the camera. All other groups in the competition made use of cardboard shelters to block out the direct light from overhead, but the color fication system performed accurately without any aids. Given more time we would integrate PRM into the system to plan more efficient paths through the workspace while avoiding obstacles.

References

[1] "Rob 550 arm lab manual."

[2] J. Z. Kolter and A. Y. Ng, "Task-space trajectories via cubic spline optimization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 1675–1682.

[3] M. W. Spong, *Robot Dynamics and Control Second Edition*, 2004.

[4] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

APPENDIX A
BILL OF MATERIALS

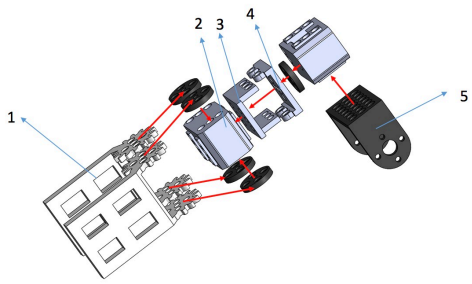


Fig. 5: Bill Of Materials

Material Name	Number	Label in the figure
3D printed clamps	2	1
XL-320 Servo motor	2	2
AX-12 Servo motor	4	N/A
6mm OLLO Revits	28	N/A
9mm OLLO Revits	4	N/A
XL320HubToXL320Base	1	5
XL320HubToOllo	3	4
XL320InterConnect	1	3

TABLE V: Bill of material for gripper.

APPENDIX B

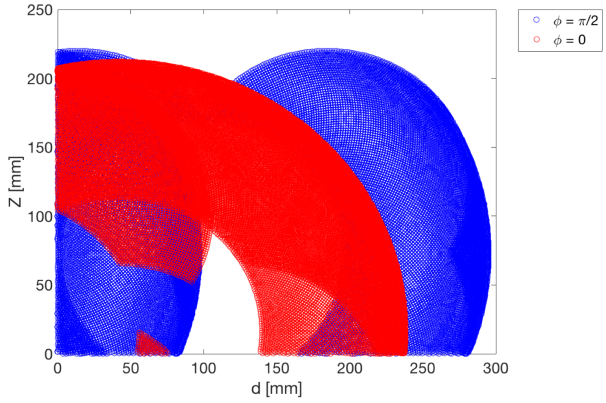
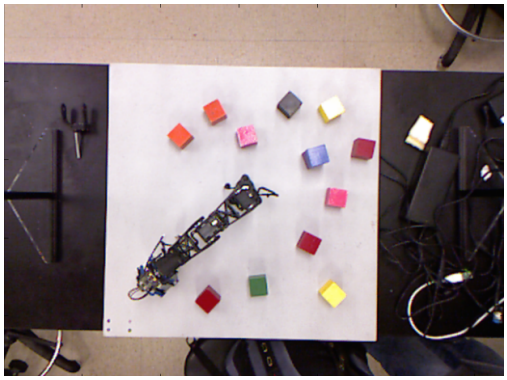
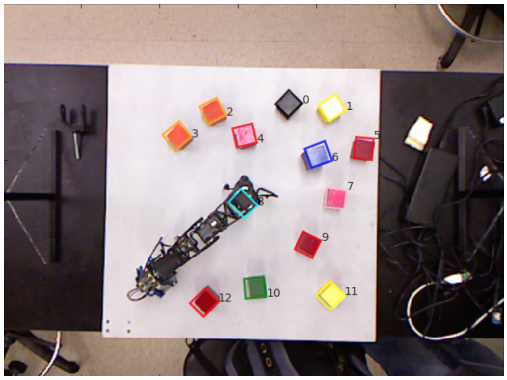


Fig. 6: Range of worskspace for a given orientation $\phi = 0$ rad and $\phi = \pi/2$ rad. "d" is the distance of end effector from the base of the arm in XY-plane and "Z" is the height of the end effector along the Z-axis

APPENDIX C
PERCEPTION SYSTEM OUTPUTS



(a)



(b)

Fig. 7: RGB image from mounted Kinect sensor and detections made by the perception system.

APPENDIX D
GRID FOR CAMERA CALIBRATION EXPERIMENT

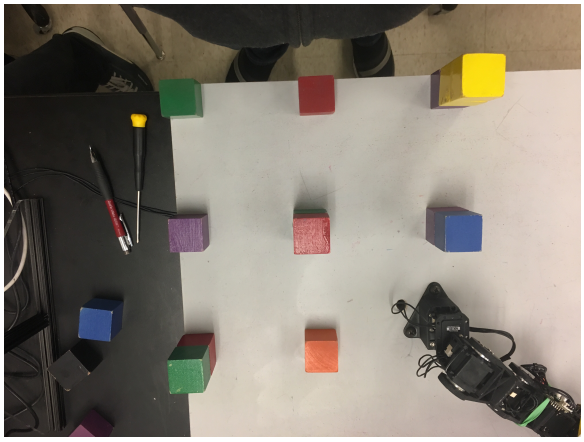


Fig. 8: Grid used to evaluate the camera calibration and block detection accuracy.

APPENDIX E
MODEL USED BY COLOR CLASSIFIER

Fig. 9: μ_c learned by color classifier for each block color. From left to right, top to bottom: black, purple, pink, red, orange, yellow, green, blue.