ROB 550 Computer Vision Report - Team 10

Pou-Chun Kung, Sarvesh Mayilvahanan, Manav Prabhakar
{pckung, smayil, prmanav}@umich.edu

***Abstract*—The primary drivers for our robotic arm are its computer vision algorithms. For different events and tasks that our team performed, camera calibration, workspace reconstruction, block detection and 3D pose estimation are essential for figuring out the arm kinematics. Here, we present the different approaches we adopted for accomplishing the vision-based goals and in turn the event tasks.**

I. Introduction

This report discusses the various aspects of the image processing pipeline used for this project. We begin with a discussion of our calibration methods. This is followed an explanation of how we use the calibration parameters along with image processing techniques to enforce workspace isolation. The workspace refers to the grey board where the blocks are placed and all operations are being performed. Anything outside the board is considered noise to our vision pipeline and we aim to discard it by reconstructing our workspace. The final step is the block detection integrated with depth information for getting the three-dimensional pose and orientation of the block with Fig. 1 shows the final result once the entire pipeline is implemented.

II. Camera calibration

A. Intrinsic Calibration

Camera calibration is an essential step to get precise 3D poses of image detected objects. In order to map the pixel coordinates to camera coordinates as a ray, we need to find the intrinsic camera parameters. Intrinsic parameters include scale factor, focal length, camera optical center, skew, and geometric distortion.

$$K = \begin{bmatrix} f_x & skew & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

K is the intrinsic calibration matrix with f_x, f_y as focal lengths and c_x, c_y as the camera optical center in the x and y dimension.

1) Methodology: Although a factory intrinsic matrix was provided with the Realsense camera, we found our own calibration using the OpenCV calibration tool and the provided checkerboards in the lab.

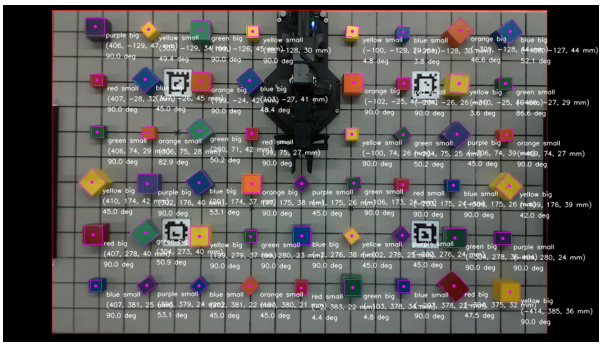

Fig. 1. Blocks detection, classification, and pose estimation results. Purple bounding boxes and dots represent the blocks' top surface and center. The first row of texts next to the blocks shows color and size classification results. The second row shows the 3D position of each block. The third row shows the estimated yaw angle of each block.

2) Results: We performed the checkerboard calibration 3 times to get three distinct intrinsic matrices and took an average of the matrices to get K_{avg}

$$K_1 = \begin{bmatrix} 896.67 & 0 & 610.22 \\ 0 & 891.81 & 397.8 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$K_2 = \begin{bmatrix} 949.97 & 0 & 639.75 \\ 0 & 957.09 & 417.68 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$K_3 = \begin{bmatrix} 969.41 & 0 & 626.96 \\ 0 & 956.59 & 429.15 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$K_{\text{avg}} = \begin{bmatrix} 938.68 & 0 & 625.64 \\ 0 & 935.16 & 414.88 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$K_{\text{factory}} = \begin{bmatrix} 904.32 & 0 & 644.01 \\ 0 & 904.82 & 360.78 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

3) Discussion: We found the self-calibrated intrinsic matrices to be fairly unstable. The maximum focal length difference is 72.74 pixels, and the maximum optical center difference is 31.35 pixels. We think this is because the checkerboard we used is not perfectly flat, and the

calibration results are sensitive to the checkerboard poses in different calibration sequences. Therefore, we chose to trust the factory calibration. The intrinsic calibration matrices evaluation can be found in Sec. II-D.

B. Extrinsic Calibration

Extrinsic camera calibration is the transformation between camera coordinates and world coordinates. We use it to find the 3D pose of detected objects with respect to the world frame.

1) Methodology: We first used physical measurements of the lab apparatus and came up with a rough extrinsic matrix. Also, we used an AprilTag bundle to get a more reliable extrinsic matrix automatically without the need for hand measurements.

2) Results: The nominal extrinsic matrix we got from measurements is

$$E_{\text{nominal}} = \begin{bmatrix} -1 & 0 & 0 & 0.025 \\ 0 & 1 & 0 & 0.135 \\ 0 & 0 & -1 & 0.975 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The extrinsic matrix we got from AprilTag bundle is

$$E_{\text{AprilTag}} = \begin{bmatrix} -1 & -0.006 & 0.021 & 0.011 \\ -0.007 & 1 & -0.02 & 0.151 \\ -0.021 & -0.02 & -1 & 0.978 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

3) Discussion: In order to get E_{nominal}, we assume the camera plane is parallel to the workspace plane, and the translation values are measured using a tape measure, which can easily cause errors. In contrast, by leveraging AprilTag detection and SolvePnP function in Opencv, E_{AprilTag} not only estimates translation values but also estimates the small rotational difference between two frames. Therefore, we believe the AprilTag bundle method provides a more reliable result. The extrinsic calibration matrices evaluation can be found in Sec. II-D.

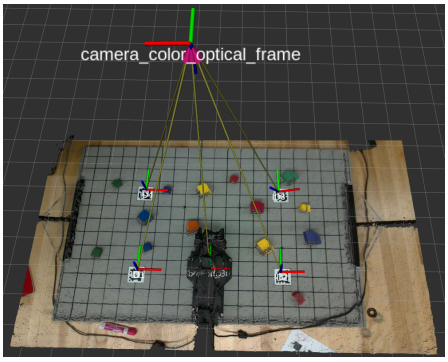

Fig. 2. Extrinsic calibration result using the AprilTag bundle method.

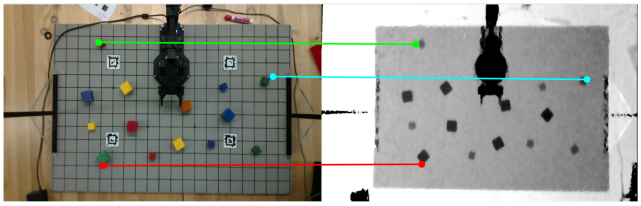

Fig. 3. Three block corners selected to find an affine transform between depth and RGB images.

C. RGB and Depth Image Alignment

We found that RGB and Depth images provided by the Realsense camera are not well aligned. Thus, we use affine transformation to register RGB and depth images.

1) Methodology: We select three points in an RGB image with known locations and select corresponding points in the depth image. Next, we use an affine transformation to register two points set to align RGB and depth camera. An illustration is shown in Fig. 3

2) Results: Fig. 4 shows the depth image before and after the alignment. Before the calibration, the color point cloud generated by depth and color images show the wrong 3d reconstruction. After the calibration, the block shapes are correct.

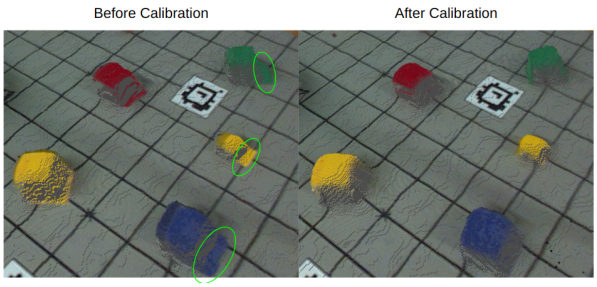

Fig. 4. A 3D reconstruction before and after depth and RGB images alignment. Lime ellipses indicate the problem caused by misalignment.

D. Camera Calibration Evaluation

To evaluate intrinsic and extrinsic calibration matrices, we use both intrinsic and extrinsic matrices and the corresponding depth image to estimate the 3D positions of a set of pixels with known positions. The evaluation image frame is shown in Fig. 5 and the labeled block centers are shown as white dots.

1) Methodology: As X_c, Y_c, Z_c denote position with respect to camera frame, and u, v denote pixel in image frame, pinhole camera projection model can be present as following.

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{9}$$

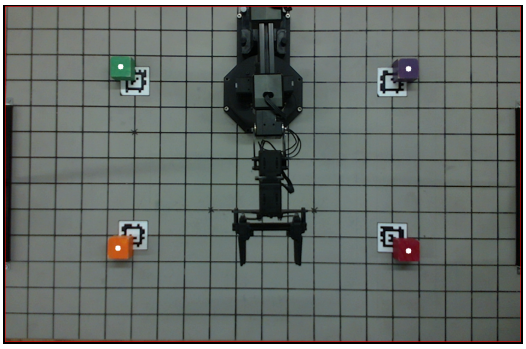

Fig. 5. The scene used to evaluate calibration matrices. The white dots represent the labeled centers of the blocks.

As Z_c can be obtained by depth image, X_c, Y_c can also be obtained by following equations.

$$X_c = (u - c_x) * Z_c / f_x \tag{10}$$

$$Y_c = (v - c_y) * Z_c / f_y \tag{11}$$

Next, extrinsic matrix is used to map camera coordinates to world coordinates.

$$E \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{12}$$

X_w, Y_w, Z_w denote position with respect to the world frame. Since the blocks in Fig. 5 are placed in known locations, we compute the translation errors of estimated block positions to evaluate different calibration matrices.

2) Results: Table. I shows the mean squared error (MSE) of block position estimation using the different intrinsic and extrinsic matrices.

Setting	Intrinsic Matric		Extrinsic Matrix		MSE (mm)
	K_{avg}	$K_{factory}$	$E_{nominal}$	$E_{AprilTag}$	
1	O		O		55.0
2	O			O	57.57
3		O	O		8.7
4		O		O	**8.58**

TABLE I
CALIBRATION MATRICES EVALUATION

3) Discussion: We found that for intrinsic matrix, factory calibration performs better than our own calibration. Additionally, the extrinsic matrix obtained by the AprilTag bundle in conjunction with the factory intrinsic matrix provided the smallest error, achieving centimeter-level accuracy for block position estimation.

III. WORKSPACE RECONSTRUCTION

The board is our primary workspace and is the region of interest, and anything outside of the bounds adds to the noise in color segmentation and detection. Thus, it was imperative to focus on methods that could isolate the board from the rest of the workspace for increased efficacy in the dependent processes.

A. Methodology

The entire process of workspace reconstruction can be divided into three major sub-processes.

1) Workspace Isolation: We use template matching to find areas of an image that are similar to a given patch referred to as the template. In our case, we take Fig. 6 as the template. This is because we aim to find the intersections of the grid lines on the board and then extrapolate the four furthest points to identify the boundary points for the board.

Fig. 6. The template image we use for matching. It is noteworthy that we intend to track all the intersection points rather than the square boxes

Image warping is another image manipulation technique that can be used to project our image to a new image plane and is facilitated by extracting boundary points from the given image which we experimented with but didn't use because it is challenging to take into consideration the intrinsic and extrinsic calibration matrices during image warping.

2) Workspace Retrieval: Once the boundary points are obtained, we retrieve the region of interest and make it as our final workspace.

After attaining the boundary points, we extract the pixels within the boundary points and pass them further down the pipeline for block detection.

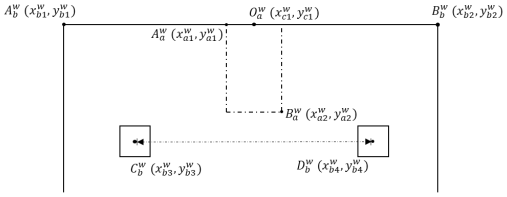

Fig. 7. The world coordinates and their integration with the image frame

3) Workspace Integration: Once the boundary points are obtained, we use the extrinsic matrix parameters along with some tested threshold values to determine the position of the arm. In an ideal scenario, the arm would be placed in the center of the image as well as the world frame. While the latter is always true, camera position

determines the former. Thus, extrinsic parameters play a pivotal role in setting the threshold values for the arm. We calculate the distances between the April Tags in the image frame which are then used for matching the distance between the board end points in the image frame and the world frame. This helps us to set the threshold values precisely.

Let k_{bi}^w and k_a^w denote the board and the arm coordinates in the world frame respectively and $k_{b_i}^i$ and $k_{a_i}^i$ denote the board and arm coordinates in the image frame respectively. Then,

$$|x_{b_1}^i - x_{b_2}^i| = \alpha |x_{b_1}^w - x_{b_2}^w| \tag{13}$$

where $\alpha \in \mathbb{R}$ provided the camera and the board lie on parallel planes. We also have

$$(x_{c_1}^w, y_{c_1}^w) = \Big(\frac{x_{b_1}^w + x_{b_2}^w}{2}, \frac{y_{b_1}^w + y_{b_2}^w}{2}\Big)$$
$$x_{c_1}^i = x_{b_1}^i + (x_{b_2}^i - x_{b_1}^i) \tag{14}$$

From Eq. 13, we get

$$x_{c_1}^i = x_{b_1}^i + \alpha |x_{b_1}^w - x_{b_2}^w| \tag{15}$$

Now we have $x_{b_1}^w$, $x_{b_1}^w$ (from physical measurements and definition of our world frame) and $x_{b_1}^i$ (from template matching) which can be used to find the coordinates for O_a^w. (For brevity, we have skipped the equations for the y-coordinates but they can be derived similarly as their x counterparts. Now, we need A_a^i and B_{ai} for separating the arm base from the rest of the region.

Let the actual arm width be W and arm length be L. Then, we have

$$W = |x_{a_1}^w - x_{a_2}^w| \tag{16}$$

$$L = |y_{a_1}^w - y_{a_2}^w| \tag{17}$$

So,

$$x_{a_1}^i = x_{c_1}^i - \frac{\alpha W}{2}$$

$$x_{a_1}^i = x_{c_1}^i - \Big[\frac{1}{\alpha} \times \frac{|x_{a_1}^w - x_{a_2}^w|}{2}\Big] \tag{18}$$

$$x_{a_2}^i = x_{c_1}^i + \Big[\frac{1}{\alpha} \times \frac{|x_{a_1}^w - x_{a_2}^w|}{2}\Big] \tag{19}$$

We can similarly calculate the value for the y coordinates of A_a^i and B_a^w.

$$y_{a_1}^i = y_{c_1}^i \tag{20}$$

$$y_{a_2}^i = y_{c_1}^i + \Big[\frac{1}{\alpha} \times \frac{|y_{a_1}^w - y_{a_2}^w|}{2}\Big] \tag{21}$$

B. Results and Discussion

Using template matching, we are able to detect the more than 90 percent of the workspace. This combined with the extrinsic calibration parameters gives us the complete workspace isolated from its background noise as shown in Fig. 5. Additionally, Fig. 1 shows the block detection once the template matching has been successfully deployed and the area outside the workspace is discarded.

IV. Block detection

To pick up the blocks, we have to recognize where the blocks are in the image. In this section, we compare three methods we implemented in order to detect block centers in an image.

A. Methodology

The first method leverages pure color segmentation. We first convert RGB images into HSV images then we use "cv2.inRange()" function to achieve color segmentation. After color segmentation, "cv2.GaussianBlur()" is deployed to smooth the color segment, then "cv2.minAreaRect()" is used to obtain a bounding box for each block. The results are shown in cyan in Fig. 8. So far, we are able to extract rough bounding boxes for blocks but still fail to find accurate block centers, because color segments include blocks' side and top surfaces.

Based on the pure color segmentation method, we proposed other two methods to extract more accurate centers of detected blocks. The first method is to incorporate depth images. Under the assumption that the center of color segmentation is located at a top surface of a block, we set an upper and lower bound for depth value in the segment to extract the top surface of a block. By extracting the surface facing upward, we can easily obtain new centers for blocks. If the color segmentation center is (u, v) with depth d, $\forall$ pixel $i \in S$ color segment if $d_i > d - \delta$ and $d_i < d + \delta$ then $i \in S'$. S' is the refined segment with only the upward-facing surface of the block. The results are shown in peach color in Fig. 8.

The second refined method is color refinement. Based on the observation that the top surface of blocks usually has different colors compared to side surfaces, and under the assumption that the center of pure color segmentation is located at the top surface of a block, we set an upper and lower bound for HSV value in the segment to extract the top surface of a block. If the color segmentation center is (u, v) with HSV value (h, s, v), $\forall$ pixel $i \in S$ color segment if $h - \delta_h < h_i < h + \delta_h$, $s - \delta_s < s_i < s + \delta_s$, and $v - \delta_v < v_i < v + \delta_v$ then $i \in S'$. S' is the refined segment with only the upward-facing surface of the block. The results are shown in purple in Fig. 8.

B. Results

First, we use the scene shown in Fig. 5 to evaluate our method. The detection results of individual blocks are shown in Fig. 8 and the quantitative analysis is shown in

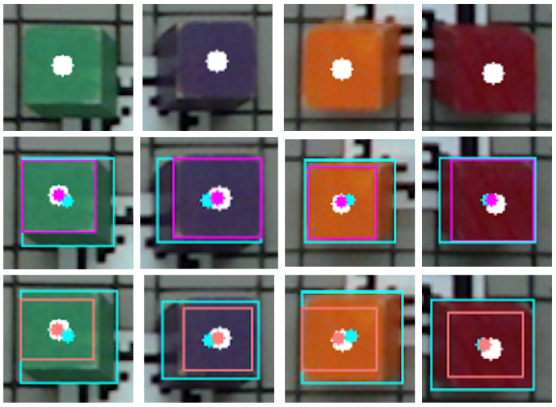

Fig. 8. Block detection ground truth and results. The white dots are labeled block centers. The cyan boxes and dots are the result from pure color segmentation. Results with depth image refinement and color refinement are shown in peach and purple color.

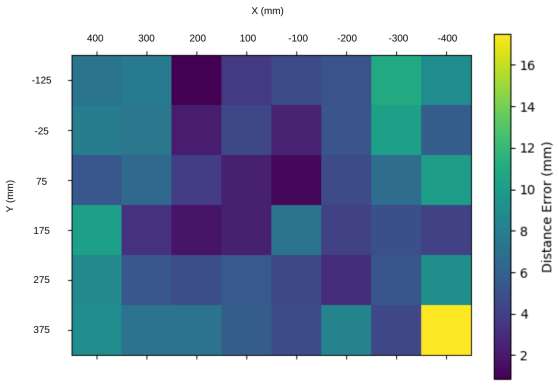

Fig. 9. 3D position estimation error distribution on the workspace. The color represent the distance error at different location.

Table. II. We also tested our method in the scene shown in Fig. 1 to analysis the relation between estimated position error and block locations, Fig. 9. Moreover, to evaluate the color classification performance we show a confusion matrix in Fig. 10.

	purple	red	greeen	orange	MSE (pixel)
ground truth	(939, 189)	(938, 520)	(414, 185)	(409, 515)	
color seg.	(935, 190)	(936, 520)	(418, 188)	(413, 515)	4.76
color seg. + color refine	(939, 189)	(938, 520)	(414, 185)	(409, 515)	**1.45**
color seg. + depth refine	(939, 194)	(938, 517)	(420, 188)	(415, 512)	1.96

TABLE II
DETECTION EVALUATION

C. Discussion

The table shows that both color refinement and depth refinement reduce errors from the pure color segmentation method. However, color refinement performs better than depth refinement. After a detailed investigation, we suggest that the main reason is that the depth image is not precise despite aligning using the method proposed in Sec.II-C.

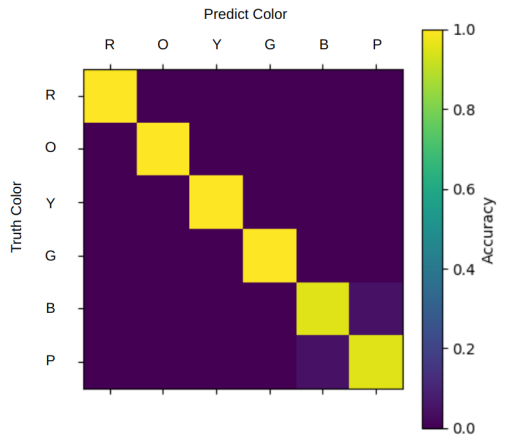

Fig. 10. Confusion matrix for color detection.

Fig. 9 shows that the position estimation is more accurate at the center of the workspace. This is because the camera distortion effect increases at the edge of the image. Fig. 10 shows that our color classification has great performance. Our block classification method only fails to classify one block in a single frame. In this case, it classifies a purple block as a blue block due to the light changing.

V. CONCLUSION

We are able to show that our image processing pipeline works well and can support our kinematics modules. The camera calibration, workspace reconstruction and block detection all work in tandem and have been optimized to provide the most optimal outputs.

There is indeed further scope for this project in implementing machine learning and deep learning approaches as they are ousting conventional image processing algorithms. However, any application of those techniques to the current use cases is highly restrained primarily because of the absence of a good quality dataset. To perform any learning tasks, we would require a large number of annotated frames with variance in block poses, camera angles, block colors and block sizes. This would require a considerable amount of manpower and time but is definitely worth testing to see potential improvements.