

Implementation of vision system on UR10e Cobot

*submitted in Partial Fulfilment of the
Requirements for the Degree of*

BACHELOR OF TECHNOLOGY IN MECHANICAL ENGINEERING

Submitted By:

Shrey D Shah (19BME134)



**DEPARTMENT OF MECHANICAL ENGINEERING,
SCHOOL OF ENGINEERING,
INSTITUTE OF TECHNOLOGY,
NIRMA UNIVERSITY,
2022**

Declaration

This is certify to that

- The project work comprises my original work towards the **Minor Project** (2ME703) subject of 7th semester in Mechanical Engineering at Nirma University and has not been submitted elsewhere.
- Due acknowledgement has been made in the text to all other materials used.

Sign:

Name: SHREY DHARMENDRA SHAH
Roll No: 19BME134

Undertaking for Originality of the Work

I, Shrey Dharmendra Shah (19BME134) give the undertaking that the Minor Project entitled "Implementation of vision system on UR10e cobot" submitted by me, towards the Minor Project (2ME703) subject of 7th semester of Bachelor of Technology in Mechanical Engineering of Nirma University, Ahmedabad, is the original work carried out by me. I give assurance that no attempt of plagiarism has been made. We understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place: Ahmedabad

**Endorsed by
(Signature of guide)**

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that, Mr. Shrey Dharmendra Shah student of Mechanical engineering, 7th Semester of, Institute of Technology, Nirma University has satisfactorily completed the project report titled “Implementation of vision system on UR10e cobot”.

Date:

Dr. Jatinkumar Dave
Guide, Associate Professor,
Department of Mechanical Engineering,
Institute of Technology
Nirma University, Ahmedabad

Prof. Vijay Ukani
Co-Guide, Associate Professor,
Department of Computer science Engineering,
Institute of Technology
Nirma University, Ahmedabad

Prof. K. M. Patel
Head and Professor,
Department of Mechanical Engineering,
Institute of Technology
Nirma University, Ahmedabad

Approval Sheet

The Project entitled **Implementation of vision System on UR10e Cobot** by **Shrey Dharmendra Shah (19BME134)** is approved towards the Minor Project (2ME703) subject of 7th semester of Bachelor of Technology in Mechanical Engineering of Nirma University, Ahmedabad

Examiners

Date: _____

Place: _____

Acknowledgement

I appreciate the chance to work on a project at the Centre for robotics and automation, which was made possible by the faculties at mechanical engineering department. For giving me the chance to work in the diverse fields at CRA, **Dr. K. M. Patel** is to be sincerely thanked.

I appreciate the technical advice and unwavering support I received from my project mentor **Dr. Jatinkumar Dave** (Associate professor, ME) and **Dr. Vijay Ukani** (Associate professor, CSE), during the course of the project.

Working with staff members at **CENTRE FOR ROBOTICS AND AUTOMATION, CRA** has been a great delight. As this project draws to a close, I can already feel that my technical understanding has grown significantly, and this experience will be crucial in determining how my aspirational technical career will develop.

I would also like to thank **Mr. Rushiraj Vala** (Research Scholar) for providing me assistance during the off-hours of University with his prowess. I take this as an opportunity to appeal my profound gratitude to all those who have directly or indirectly help me in my venture.

Shrey Dharmendra Shah

Date:

Place:

Abstract

Robots equipped with a vision system holds immense potential in aiding humans, making their lives easier. This project aims the same for neurologically challenged patients. The UR10e cobot is a 6-DOF cobot with detachable and variable end-effectors. The robot has a workspace of 1.3 meters. The robot will be connected with a camera taking images of the work table. The dynamic vision system captures objects and passes on the coordinates and orientation to the robot, which then picks up the recognized object.

Several attempts are made for pick and place applications but most do not rely on vision systems for the application rather the desired object location is entered manually by the operators in real time whereas those that incorporated a vision system limit the objects to black or white colours for easier identification. This project hopes to accomplish a more capable vision system along with connecting it with the UR10e cobot.

Images taken from the live feed were processed and several algorithms applied to remove the noise from the detection areas. The absence of ArUco markers in matlab further facilitated a need for reference points for precision. Robot operating system established a connection with the robot enabling the PC to control it as desired.

Table of Contents

1. Introduction	10
1.1. Objectives and problem statement	10
1.2. Matlab	10
1.3. UR10e	11
2. Literature Review.....	11
3. Methodology	16
3.1. Methods used throughout the Project	17
3.1.1. Canne Edge Detection	18
3.1.2. Gaussian Filter.....	19
3.1.3. Dilation	19
3.1.4. Erosion	20
3.1.5. Circularity.....	20
3.1.6. FK and IK	21
3.2. Tools used.....	24
3.2.1. Experimental Setup.....	24
3.2.2. UR10e	25
3.2.3. Matlab	26
3.2.4. Ubuntu.....	26
3.2.5. ROS.....	27
4. Results and Discussions	28
4.1. Yellow filter.....	28
4.2. Reference filters for getting corner points	30
4.3. Detecting Rectangles and major minor axes.....	31
4.4. ROS and Rviz	34
5. Conclusion and Summary	39
5.1. Future scope.....	39
6. References	40

List of Figures

FIGURE 1 - APPLICATIONS OF VISION SYSTEMS	10
FIGURE 2 - UR10E HOME POSITION	11
FIGURE 3 - DATA AND END - EFFECTOR	12
FIGURE 4 - COLOUR SEGMENTATION	12
FIGURE 5 - EXPERIMENTAL SETUP WITH ARUCO	13
FIGURE 6 - FORWARD KINEMATICS WITH LOCATION 2DOF	14
FIGURE 7 - LIVE TRACKING USING PROJECTIVE TRANSFORMATION	14
FIGURE 8 - LASER DETECTION	15
FIGURE 9 - COMPONENT CLASSIFICATION	16
FIGURE 10 - FLOWCHART	17
FIGURE 11 - DILATION OF IMAGE	19
FIGURE 12 - EROSION OF IMAGE	20
FIGURE 13 - CIRCULARITY COMPARISON	21
FIGURE 14 – 2D FRAME REFERENCE FOR UR10E	22
FIGURE 15 - 3D FRAME REFERENCE FOR UR10E	22
FIGURE 16 - TRANSFORMATION MATRIX CALCULATIONS	23
FIGURE 17 - CRA EXPERIMENTAL SETUP	24
FIGURE 18 - TEACH PENDANT OF UR10E	25
FIGURE 19 - UR10E USED FOR PROJECT	26
FIGURE 20 - DISK PARTITION FOR UBUNTU	27
FIGURE 21 - ROS RUNNING SAMPLE	28
FIGURE 22 - COLORIZATION	29
FIGURE 23- ISOLATING THE BLOCKS	29
FIGURE 24 - NORMAL PLACEMENT	29
FIGURE 25 - OVERLAPPING CONDITIONS	30
FIGURE 26 - HANGING CONDITION	30
FIGURE 27 - CORNER DETECTION (A)	31
FIGURE 28 - CORNER DETECTION (B)	31
FIGURE 29 - RECTANGLE DETECTION (A)	32
FIGURE 30 - RECTANGLE DETECTION (B)	33
FIGURE 31- COMPLETE RESULTS (A)	33
FIGURE 32 - COMPLETE RESULTS (B)	34
FIGURE 33 - CALIBRATION TESTING	35
FIGURE 34 - ROSLAUNCH WITH MOVEIT!	35
FIGURE 35 - RVIZ DEMONSTRATION	36
FIGURE 36 - EXAMPLE WITH OPERATING ROBOT	37
FIGURE 37 - SIMULATION OF EXAMPLE IN RVIZ	37
FIGURE 38 - ROBOT MANOEUVRING VIE MOVEIT! AND RVIZ	38

1. Introduction

In recent years, Object recognition has been the main focus of the computer science community. The problem of using singular deep learning algorithms for classification and identification of objects is apparent. It utilizes Convolution neural networks (CNN) which requires large amounts of RAM long with high specs of the host device. Although it isn't overinflated, the use of such devices incurs huge costs when operated collectively. The use of image processing reduces the time and calculations required for object identification. By using geometrical features of the test objects, this method removes the disadvantages of sole deep learning. Filtered data from image processing is fed to advanced vision systems for further classification. The application of such image processing techniques of computer vision can be found everywhere in day-to-day life.



Figure 1 - Applications of vision systems

1.1. Objectives and problem statement

Recognize objects based on shape regardless of colour and use the UR10e cobot to deliver the object to the desired person or location as specified by the user before initializing, the robot aims to help neurologically challenged patients or in areas where human interaction is unfeasible such as a pandemic.

1.2. Matlab

Matlab is a proprietary multi-paradigm tool used for various programming applications. The project used MATLAB for all logical algorithms and image processing facilities, the R2022b version was used with several packages and dependencies for completing the project.

Visualization of resulting images was showcased in Matlab's batch processing app and graphical interfaces for initial simulations of the robot.

1.3. UR10e

The robot used in this project can be manipulated in 6 different degrees of freedom. Basic plane restriction conditions are set in wrist_4 and wrist_5 for collision avoidance. 2 more plane restrictions are set on the workspace in the xy-plane and the plane 45° with the zy-plane. The robot has a gripper installed with maximum extension of 6 cms along with a combined payload of 12 kilos. The robot is set 45° with the table workspace facing -y direction as shown in figure 2.

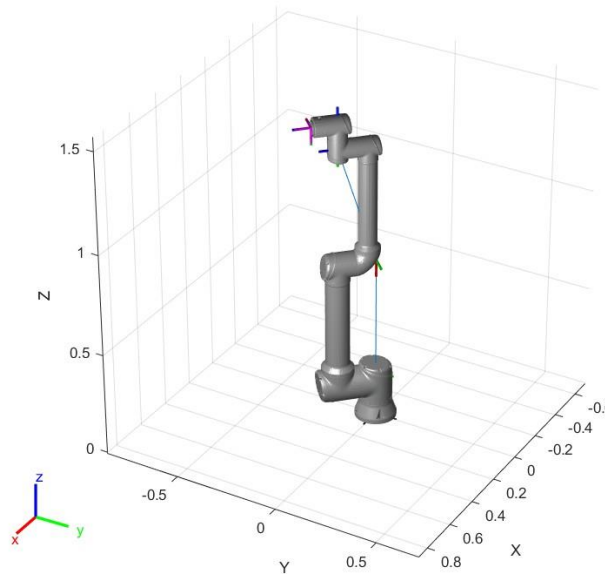


Figure 2 - UR10e home position

2. Literature Review

Vision Based Object Classification with Scorobot ER-4U [1]

This paper focuses on using vision system to find the object with the biggest area in a given snapshot. A wireless camera is installed at the head of the gripper for fixed frame of reference for the robot. The paper used image thresholding in a gray scale image converted from RGB colorspace. The sample objects were limited to dark coloured objects which when converted to gray scale image equalizes above a certain threshold. Centroids of the images were then found based on the true area (all 1's in binary image) after thresholding. The coordinates of the centroid are then transferred to the scorobot via the Advanced control language (ACL) interface

converting the coordinates to 8 bit configuration. The robot is then simulated in the designated Robotalk available for the scorobot.

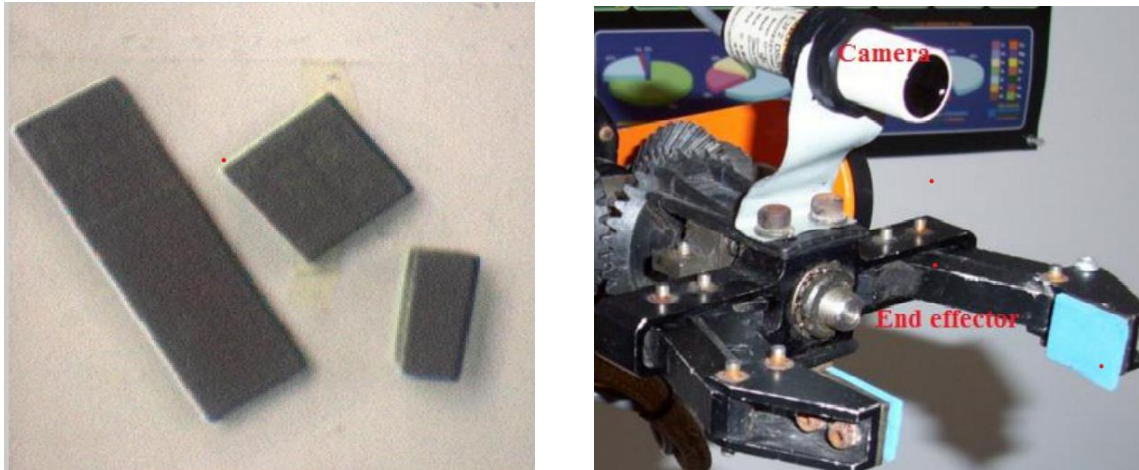


Figure 3 - Data and End - effector

Vision-based Robot Manipulator for Industrial Applications [2]

This paper focuses on object classification and arranging them based on their colour. Several colour spaces have been employed by the paper to increase efficiency of classification. This implementation does not explicitly identify what the colour of the object is rather it segregates all the object with same property (colour in this case) and assigns a number or colour of its own. Since the user need not want specific coloured-objects, the code works seamlessly. The results dictated that Lab colour space result in highest segregation of the coloured objects. Like previous paper this also uses basic thresholding then each range is assigned a specified colour or number similar to K-nearest neighbours (KNN) algorithm.

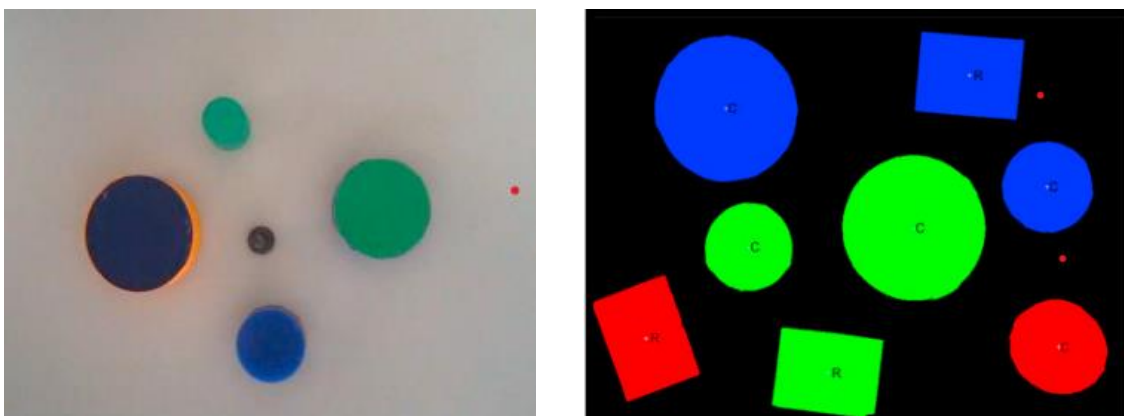


Figure 4 - Colour Segmentation

Estimation of the Gripping Position and Orientation of Fasteners in Camera Images [3]

The primary focus here is identification of orientation of long objects. ArUco markers are used as reference with python to extrapolate the boundary of the scene. This was attained by projective transforming the curved workspace to a flat 2D with borders as ArUco markers. Here a mask is used to subtract the background with the foreground keeping all the remaining objects intact. Equidistant points are carved on the base for distance and angle references. Furthermore, R-CNN is employed to classify if the object is nail, pin or a screw.

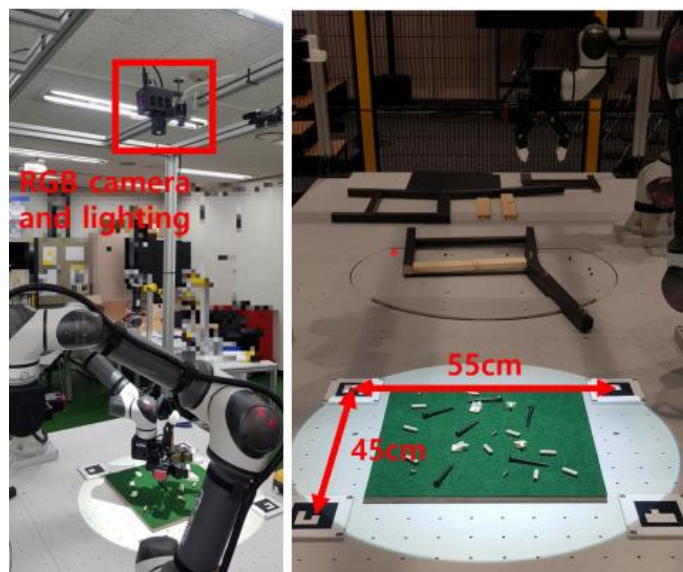


Figure 5 - experimental setup with ArUco

Vision System via USB for Object Recognition and Manipulation with Scorbot-ER 4U [4]

Here, Manual Forward kinematics is performed to reach the object identified by the camera. No reference points are used in this paper rather the location of objects is derived from the base reference link of the scorobot itself. No attempts have been made for orientation detection as aimed objects are cylindrical. Matlab Toolbox for Inteliteck scorobot (MTIS) is employed for directly controlling the bot. Similar to previous paper the background is subtracted leaving the objects on the work table resulting in easy location.

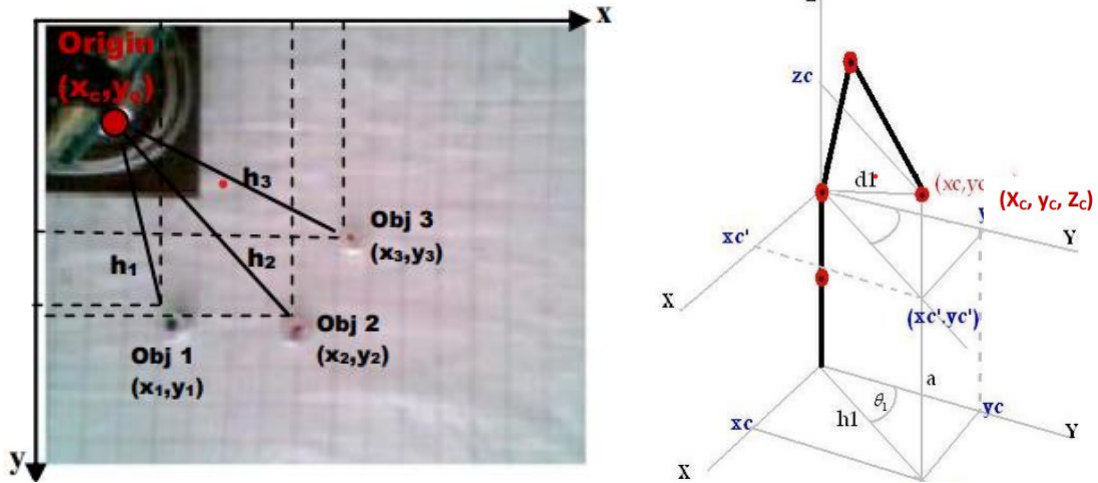


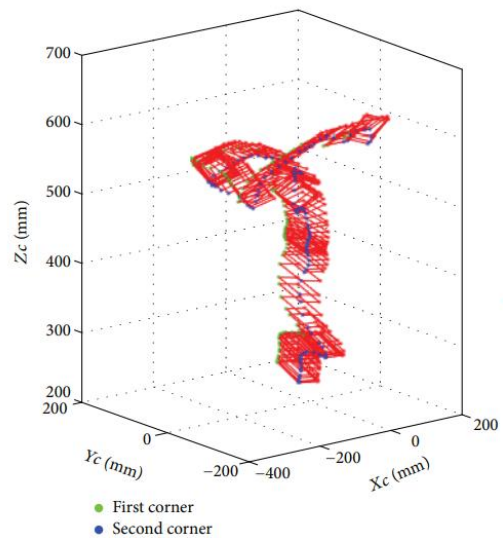
Figure 6 - Forward kinematics with location 2DOF

Vision-Based Object Recognition and Precise Localization for Space Body Control [5]

Stereo vision is employed where same geometric features are to be extracted to construct a 3D visualization of the object. This paper aims to detect inanimate objects placed at various distances with 2 different cameras acquiring the complete location in 3D cartesian space. Projective transformation is used for ArUco markers that can identify angular tilt of the object in all 3 axes. The live feed installed in this setup enables tracking of the said object in real time generating the trajectory of the object relative to the stereo cameras.



Figure 7 - Live Tracking using projective transformation



Simultaneous Object Recognition and Position Tracking for Robotic Applications [6]

Incorporating laser sensors to reconstruct a fixed trapezoidal object in a manufacturing facility, this paper also utilizes point detection in conjunction with projective transformation of the known object. The manufactured product has holes carved on them at fixed distances which does not enable the laser to reflect back to the sensor thus indicating that there exist a cavity. 2 different laser sensors in opposite directions enable the program to recreate the object using the deconstructed laser feedback.

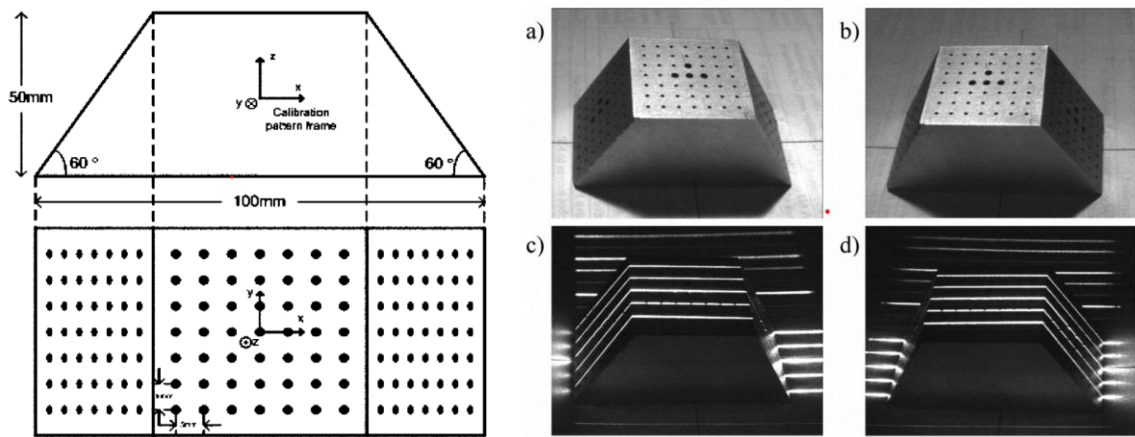


Figure 8 - Laser detection

Object Detection and Recognition for a Pick and Place Robot [7]

This paper uses the concepts of identifying contours of a sample image whose contrast has been adjusted to sharpen the edges. The contours are then used to create bounding boxes around the objects in original image. The different types of capacitors and resistors are identified by training them with a model using convolutional neural networks (CNN). Noise can occur if objects are overlapping each other also resulting in misclassification.

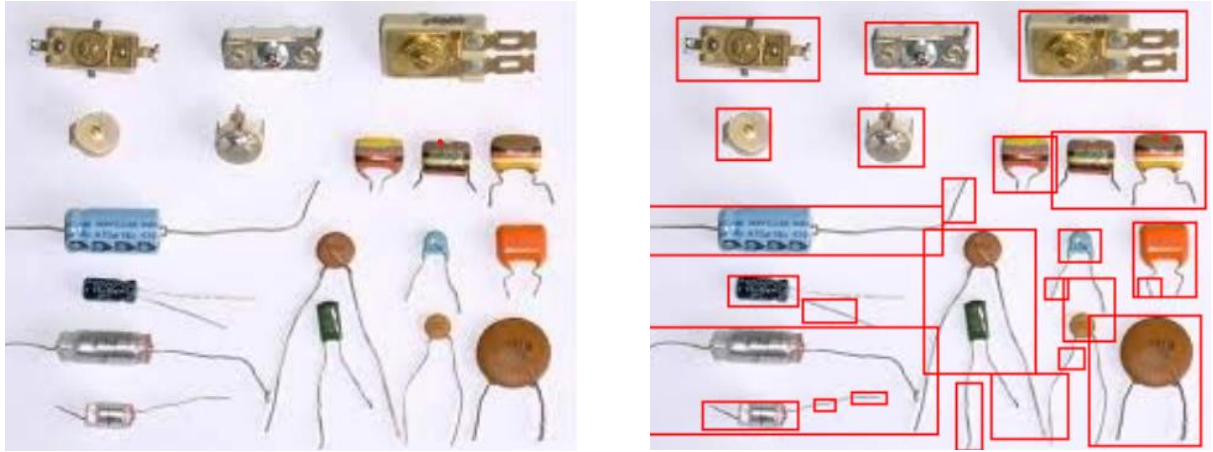


Figure 9 - Component classification

3. Methodology

A camera will be used for 2D geometry acquisition neglecting the z axis for simplicity purposes. The image will be subject to various image processing and segmentation methods. Filtering out unnecessary data from the original image. After singling out necessary objects, the features like centroids and orientation will be extracted via further calculations. Robotic operating system will be used as an interface to connect the robot to the PC via the ethernet cable passing the joint angles to UR10e. The main tasks are listed below –

- A. Color segmentation of 3 sample objects after gray scale conversion.
- B. Segmentation of all and any rectangular objects regardless of color.
- C. Forward and inverse kinematics solution for cartesian and joint angle values.
- D. Connecting ROS with UR10e to move the robot to the desired location in task space.

The flow of Research is presented below:

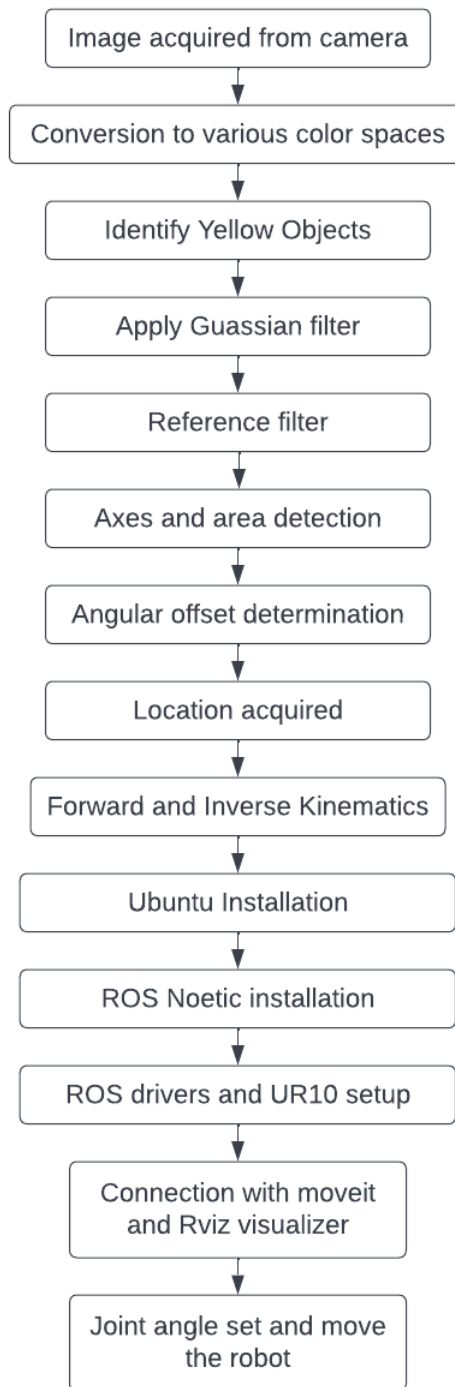


Figure 10 - Flowchart

3.1. Methods used throughout the Project

The image processing procedure requires several trial-and-error algorithms and methods for one to get the desired results. The methods presented below are used in one way or another to get the results from the image or the robot. One may change parameters of these methods

according to their liking but the results may drastically change following the butterfly effect. Hence an integration of deep learning is necessary to further solidate the results.

3.1.1. Canny Edge Detection

The canny edge method works on the principle of detecting gradients in the pixels of images segregating strong and weak gradients. The primary 4 steps of canny edge detection are stated below.

- a) Smoothing – the blurring of image is done to reduce unnecessary noise such as particles. The degree of blurring is based on standard deviation of the gaussian filter discussed below further below
- b) Gradient construction – each pixel is assigned a gradient value in the x and y directions. The combined form indicated the direction of movement of the pixel to the next cluster of pixels the. The gradient values are governed by the Euclidian distance and are given by:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (3.1)$$

$$|G| = |G_x| + |G_y| \quad (3.2)$$

Where $|G_x|$ and $|G_y|$ are gradients in the x and y direction. Since the direction of the gradient also needs to be stored for non-maximum suppression method it is stored in theta:

$$\theta = \arctan^{-1} \left(\frac{|G_y|}{|G_x|} \right) \quad (3.3)$$

- c) Non-maximum suppression – the purpose of initial blurring was to preserve the local maxima which is used to find out sharp and weak edges where the weak ones are suppressed. Double thresholding is then employed which is basically just segregating at a new value for this new image.
- d) Edge tracking by hysteresis – Here all the strong pixels are connecting via BLOBS. A BLOB is a collection of surrounding 8 pixels out of which one strong edge is selected and rest weak are discarded thus only leaving the strong ones eventually becoming the edges present in the image.

3.1.2. Gaussian Filter

Gaussian filter is basically used to blur the images or objects. The basic one-dimensional gaussian function is used everywhere and which we commonly known as the bell curve. Here the gaussian basically works as a point spread function where the mean is assumed to be zero and the value of standard deviation is set by the user. Higher standard deviation means more values towards the centre and less at the edges this creates a lack of data thus blurring the image as a result by discrete approximation for storing and comparing with the individual pixels. The gaussian function for 1D and 2D data is given below:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (3.4)$$

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.5)$$

3.1.3. Dilation

The dilation process is a moving neighbourhood process where the values of pixels are assigned based on the maximum value of the structuring element. The type of structuring element used significantly affects the values. in this project various elements are used depending on the results. Some of them are diamond, circle and square. Since our image at this stage would mostly be binary image of 1's and 0's, the result of dilation would be an increase in the areas of the true values detected. Other way to say this would be the lines get thicker by this process.

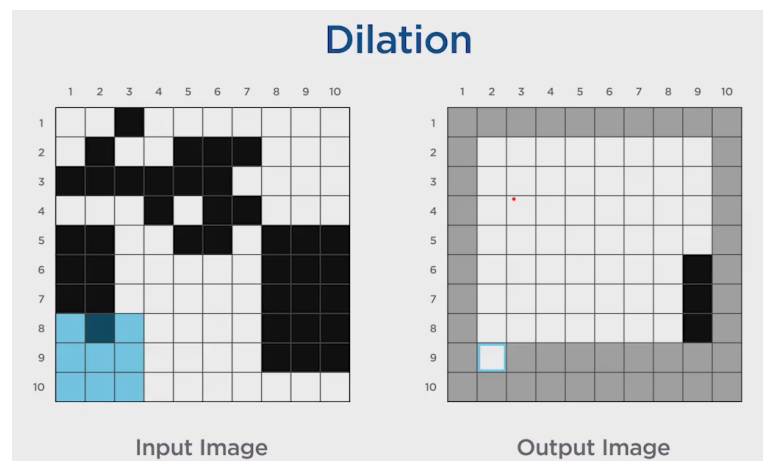


Figure 11 - Dilation of image

3.1.4. Erosion

The erosion process is the opposite to the dilation where the minimum element is selected from the structural element. The result of this process is the reduction of the areas of the true values. In erosion the lines in binary image get thinner.

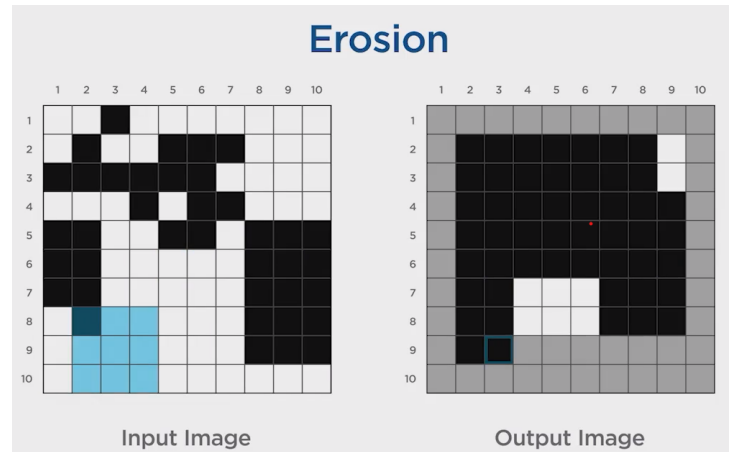


Figure 12 - Erosion of image

The process of sequentially performing dilation and then erosion is called morphological closing where holes are filled in broken images and if erosion is followed by dilation, then it is called morphological opening widening the holes of the image.

3.1.5. Circularity

Circularity is a property of shaped which tells how close the object is to being a circle. A circle has a circularity of 1 and as the shape of the object becomes irregular the circularity decreases. The formula for determining circularity is given by:

$$Circularity = \frac{Perimeter^2}{4\pi \times Area} \quad (3.6)$$

The perimeter of the true objects is determined by the number of pixels surrounding the ones. The perimeter in terms of pixels becomes slightly larger than what it actually may be, the area of the object could be determined by the number of true pixels in that closed region. The area and perimeter brought by this method me be slightly different from actual object. To determine the circularity of rectangles, a piece of code was written to plot circularities of all types of rectangles including irregular ones with length 500 and breadth 1. From 10 to 100 to 1000 and finally 10000, we can clearly see the trend that most of the rectangles lie in the range of 0.6 to 0.8. But as said previously, the perimeter is slightly larger than the actual resulting in lower

value of circularity. For this reason, we increase the circularity range for this project to 0.55 to 0.8.

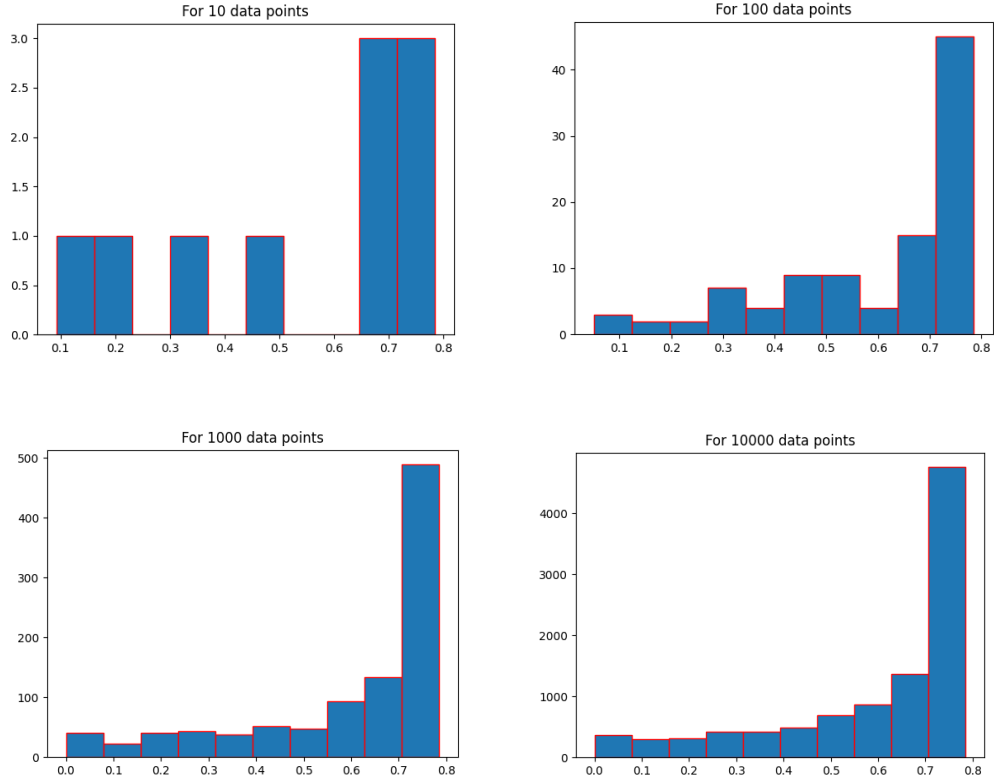


Figure 13 - Circularity comparison

3.1.6. FK and IK

The universal robot is a 6 degree of freedom (DOF) robot which has 7 reference frames including the end-effector. Forward kinematics of a robot is finding the end position of the tool along with orientation. The transformation matrix consists of a 3x3 rotation matrix and a 3x1 position vector. This matrix is used for representing coordinates in one plane in another. The formula for transformation matrix is given by:

$$T_i^{i-1} = T_{tran}(\alpha_i, 0, 0)T_{rot}(x_{i-1}, \alpha_i)T_{tran}(0, 0, d_i)T_{rot}(z_{i-1}, \theta_i)$$

$$= \begin{bmatrix} C_{\theta_i} & -S_{\theta_i} & 0 & a_i \\ C_{\alpha_i}S_{\theta_i} & C_{\alpha_i}C_{\theta_i} & -S_{\alpha_i} & -d_iS_{\alpha_i} \\ S_{\alpha_i}S_{\theta_i} & S_{\alpha_i}C_{\theta_i} & C_{\alpha_i} & d_iC_{\alpha_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

$$T_6^0 = T_1^0 \times T_2^1 \times T_3^2 \times T_4^3 \times T_5^4 \times T_6^5 \quad (3.8)$$

Where α_i , θ_i , d_i and a_i are called the Denavit-Hartenberg (DH) parameters which govern the allocation of x, y and z axis when determining the next frame with respect to the previous. Each frame is mapped out and we start transforming from the base. Each transformation is multiplied with the previous one to keep getting reference with the base until we reach the end effector.

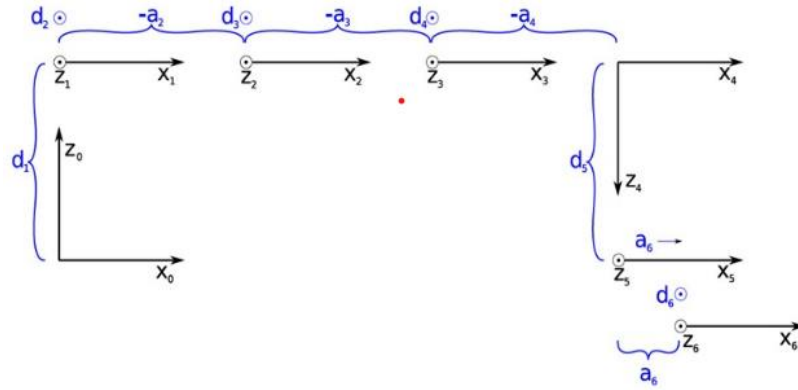


Figure 14 – 2D frame reference for Ur10e

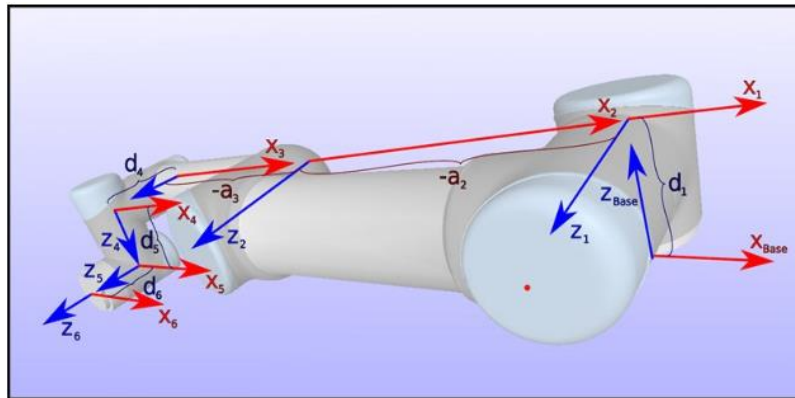


Figure 15 - 3D frame reference for UR10e

A excel file has been created for easier manipulation of forward kinematics. For simplicity purposes only Forward kinematics solution has been calculated. For the inverse kinematics solution, the transformation matrix obtained from the FK is passed on to the official IK solution identifier provided by the robotic systems toolbox. As we know inverse kinematics has more than one solution when solved by Jacobian method, hence multiple runs often produce infeasible results. A loop is employed which checks all values of joint angles are within permissible values and ends when it is satisfied. A snippet of forward kinematics solver along with the DH parameter table has been provided below:

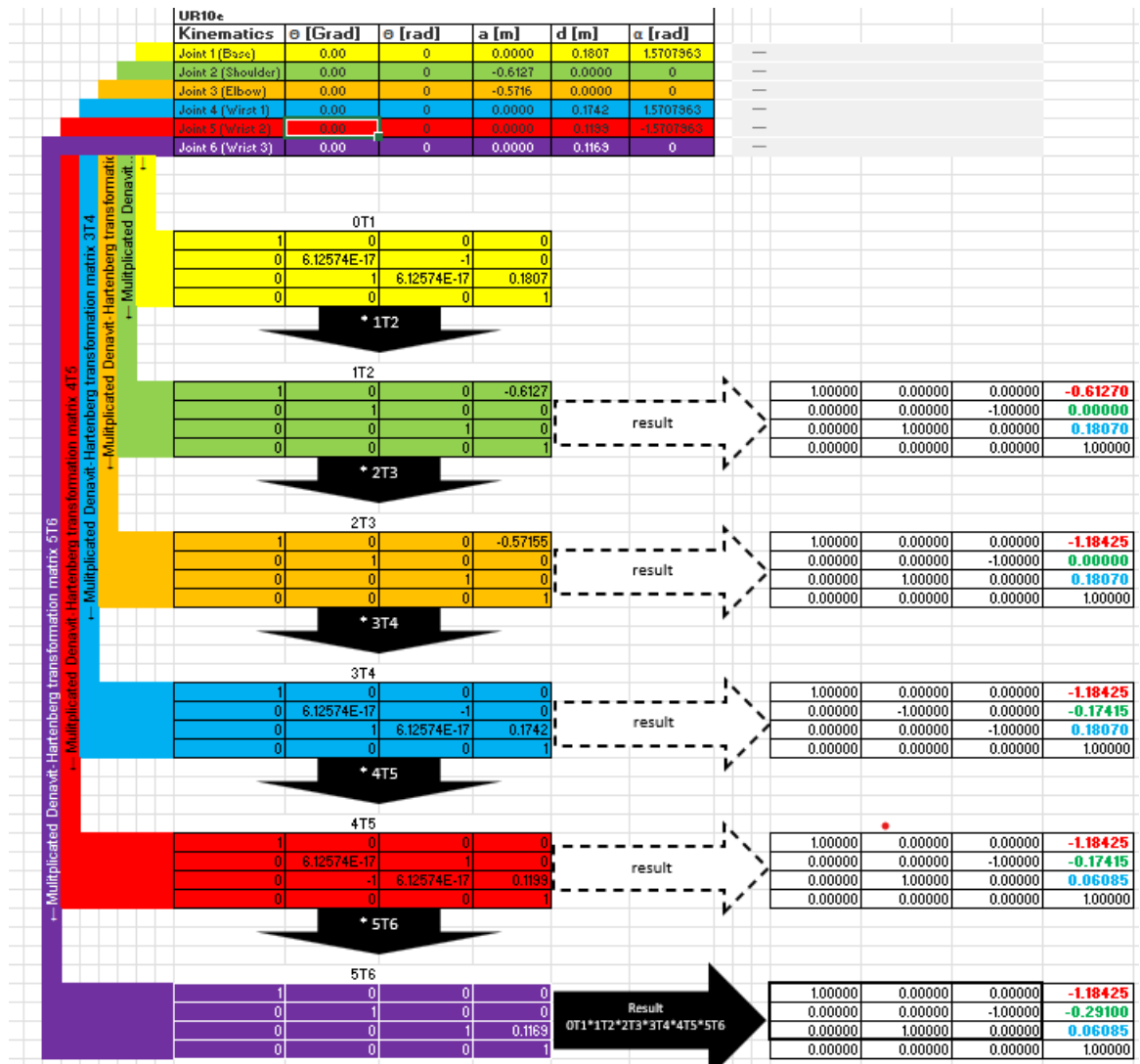


Figure 16 - Transformation matrix calculations

Table 1 – DH parameters of UR10e

Kinematics	θ , Theta (rad)	a (m)	d (m)	α , Alpha (rad)
Joint 1	0	0	0.1807	$\pi/2$
Joint 2	0	- 0.6127	0	0
Joint 3	0	- 0.5715	0	0
Joint 4	0	0	0.1741	$\pi/2$
Joint 5	0	0	0.1198	$-\pi/2$
Joint 6	0	0	0.1165	0

3.2. Tools used

For the working of this project, one needs the following to have complete control over the robot's movement and live tracking. Some of the tools are custom made and some are ready to use software or products. Installation of most packages is bothersome as the community is open source leading to no formal or updated documentation.

3.2.1. Experimental Setup

In this project, for initial image acquisitions a wooden stand was used which has a attached camera on it's head. The camera used is HP W200 for 2D vision. The test table is black coloured and measures 100 cm x 70 cm. light hinderance is present while performing after light. Hene the dataset for all image processing was taken from complex environments and everyday life backgrounds increasing the difficulty of detection. The properties of the camera along with initial setup are given below.



Figure 17 - CRA experimental setup

Table 2 – Camera Details

General Information	
Webcam type	Dedicated
Model	W200
Brand	HP
Lens Details	

Field of view	72-degree wide angle view
Connectivity details	
USB	Yes, Yes
Key features	
Resolution	720p 30 fps high resolution (max)
cord	UVC, Plug and play
Movement capabilities	270-degree swivel possible

3.2.2. UR10e

The UR10e belongs to the Centre for Robotics and Automation (CRA), Nirma University. The base axis of this robot and basic specification were extracted. From the university it was found that the axes of the bot was tilted by 45° counter clockwise. To correct the same while calculating, rotation about z by the same value is performed before sending the coordinates to the machine. The UR10e comes with a teach pendant module which controls te robot on its own. The Teach pendant is equipped with safety measures, plane restrictions and ethernet connection. For basic controlling only teach pendant can be used but for industrial application it is necessary to have some interface with PCs. The teach pendant and the robot is shown below.

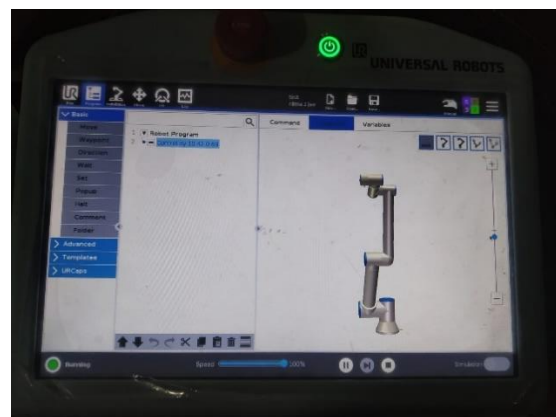
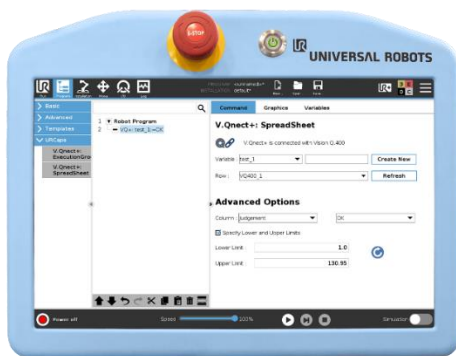


Figure 18 - Teach pendant of UR10e



Figure 19 - UR10e used for project

3.2.3. Matlab

Matlab is a crucial software in this project where all the image processing takes place. Important drivers and tool boxes such as Robotics systems toolbox, ROS toolbox and UR series support packages are needed to be installed with the main Matlab. The project uses Matlab 2022b. hardware setup for ROS host machine is also required at this stage or after downloading ROS properly. Image processing toolbox along with computer vision toolbox is also necessary for this project.

3.2.4. Ubuntu

The ROS works only on Ubuntu operating systems. Although there are open-source solutions for making it work on windows, it isn't recommended nor is virtual machine which puts a lot of loads on the machine. Ubuntu 20.04.5 (Focal Feta Release) is downloaded for better

compatibility with ROS. The main disk either needs to be partitioned or new disk added to boot Ubuntu in. I added a new SSD to dual boot entire ubuntu in it after partitioning it according to my needs, the disk 0 corresponds to the entire Ubuntu operating system.



<div> Disk 0</div> <div>Basic 465.75 GB Online</div>							
	223.52 GB Healthy (Primary Partition)	223.52 GB Healthy (Primary Partition)		18.71 GB Healthy (Primary Partition)			
<div> Disk 1</div> <div>Basic 953.85 GB Online</div>							
	260 MB Healthy (E)	OS (C:) 310.22 GB NTFS Healthy (Boot, Page File, Cr	Data_2 (E:) 310.22 GB NTFS Healthy (Basic Data Partitio	Data_1 (D:) 310.22 GB NTFS Healthy (Basic Data Partitio	750 MB Healthy (Rec	RESTORE (F:) 22.00 GB NTFS Healthy (Basic Data F	200 MB Healthy (R

Figure 20 - Disk Partition for Ubuntu

3.2.5. ROS

Robotic operating system is an open-source robotics control system. This helps one build the nodes needed to control the tasks of robots in specific topics or logical decisions. ROS Noetic was installed compatible with Ubuntu 20.04 release. The keys were imported and basic config setup from various distributed sources. The Matlab needs to work with ROS hence specific ROS drivers are also installed after lot of effort. Configuring the ip of host with the target machine and external control URcap file was installed in the teach pendant to allow for other devices to communicate with it. The main purpose is to host a ROS master which establishes a connection with the robot. ROS actively receives live location and connection states of the robot. The nodes then deliver tasks such as moving, shutting down and gripping. Matlab recognises the ROS's connection attempts and connects to the ROS server further taking in the inputs from UR10e. The ROS then operates as a channel of communication between Matlab and teach pendant.

```

ryuzaki@ryuzaki-ROG-Strix-G513RC-G513RC:~/catkin_ws$ roslaunch ur_robot_driver ur10e_bringup.launch
[INFO] [1670677679.349673813]: checksum: [2807061466 2805421401 2925280869 2435617020 2436375967 243
dh_theta1 [-4.80896718977197e-08 -0.386317268006271 7.44747281261952 -0.777967759101868 -3.133883179
88552e-06 7.91589773764534e-08 ]
dh_a: [-8.58620446614406e-05 -0.50726881561012 -0.407030386183929 -1.0875471777885e-05 -9.7391289187
3194e-05 0 ]
dh_d: [0.180894444516596 56.3186399214209 163.503203467844 -219.648404676292 0.119860088034 0.115727
912592615 ]
dh_offset: [1.5711031040838 -0.00469712234560062 0.00182432291799164 1.57082294201797 -1.5717493619
468 0 ]
calibration_status: 2
[INFO] [1670677679.449979805]: Writing calibration data to "/home/ryuzaki/my_robot_3_calibration.ya
ml"
[WARN] [1670677679.450086201]: Output file /home/ryuzaki/my_robot_3_calibration.yaml already exists
- overwriting.
[INFO] [1670677679.451048105]: Wrote output.
[INFO] [1670677679.451324184]: Calibration correction done
[calibration_correction-1] process has finished cleanly
log files: /home/ryuzaki/.ros/log/6c10bdc6-78b2-11ed-90a5-59fce789477/calibration_correction-1*.log
all processes on machine have died, roslaunch will exit
shutting down processing monitor...
... shutting down processing monitor complete
done
ryuzaki@ryuzaki-ROG-Strix-G513RC-G513RC:~/catkin_ws$ roslaunch ur_robot_driver ur10e_bringup.launch
robot_ip:=10.42.0.254 kinematics_config:=$(HOME)/my_robot_calibration.yaml

[INFO] [1670677731.778569993]: compiled against Qt version 5.12.8
[INFO] [1670677731.778580723]: compiled against Gaze version 1.9.0 (Chadamon)
[INFO] [1670677731.786954349]: Forcing OpenGL version 0.
[INFO] [1670677731.975942531]: Stereo is NOT SUPPORTED
[INFO] [1670677731.976038959]: OpenGL device: AMD YELLOW_CARP (DRM 3.42.0, 5.15.0-56-generic, LLVM
12.0.0)
[INFO] [1670677731.976085474]: OpenGL version: 4.6 (GLSL 4.6) limited to GLSL 1.4 on Mesa system.
[INFO] [1670677731.243152653]: Loading robot model 'ur10e_robot'...
[WARN] [1670677731.272448101]: IK plugin for group 'manipulator' relies on deprecated API. Please i
mplement initializeRobotModel(...)
[INFO] [1670677731.275521630]: IK Using joint shoulder_link -6.28319 6.28319
[INFO] [1670677731.275544933]: IK Using joint upper_arm_link -6.28319 6.28319
[INFO] [1670677731.275554935]: IK Using joint forearm_link -3.14159 3.14159
[INFO] [1670677731.275563678]: IK Using joint wrist_1_link -6.28319 6.28319
[INFO] [1670677731.275572223]: IK Using joint wrist_2_link -6.28319 6.28319
[INFO] [1670677731.275580609]: IK Using joint wrist_3_link -6.28319 6.28319
[INFO] [1670677731.275580205]: Looking in common namespaces for param name: manipulator/position on
ly_ik
[INFO] [1670677731.276555297]: Looking in common namespaces for param name: manipulator/solve_type
[INFO] [1670677731.277750211]: Using solve type Distance
[INFO] [1670677731.333990069]: Starting planning scene monitor
[INFO] [1670677731.338958494]: Listening to '/move_group/monitored_planning_scene'
[INFO] [1670677731.797428563]: Constructing new MoveGroup connection for group 'manipulator' in nam
espace ""
[INFO] [1670677736.931538832]: Ready to take commands for planning group manipulator.
[INFO] [1670677775.96047288]: ABORTED: Solution found but controller failed during execution

[INFO] [1670677697.357077013]: Robot's safety mode is now NORMAL
[INFO] [1670677697.430108]: Controller Spawner: Waiting for service controller_manager/switch_control
ler
[INFO] [1670677697.435541]: Controller Spawner: Waiting for service controller_manager/unload_control
ler
[INFO] [1670677697.438022]: Controller Spawner: Waiting for service controller_manager/switch_control
ler
[INFO] [1670677697.438673]: Loading controller: pos_joint_traj_controller
[INFO] [1670677697.443171]: Controller Spawner: Waiting for service controller_manager/unload_control
ler
[INFO] [1670677697.447226]: Loading controller: joint_state_controller
[INFO] [1670677697.471992]: Loading controller: joint_group_vel_controller
[INFO] [1670677697.476908]: Loading controller: scaled_pos_joint_traj_controller
[INFO] [1670677697.484220]: Controller Spawner: Loaded controllers: pos_joint_traj_controller, joint
_group_vel_controller
[INFO] [1670677697.586032]: Loading controller: speed_scaling_state_controller
[INFO] [1670677697.513791]: Loading controller: force_torque_sensor_controller
[INFO] [1670677697.521914]: Controller Spawner: Loaded controllers: joint_state_controller, scaled_po
s_joint_traj_controller, speed_scaling_state_controller, force_torque_sensor_controller
[INFO] [1670677697.529221]: Started controllers: joint_state_controller, scaled_pos_joint_traj_contro
ler, speed_scaling_state_controller, force_torque_sensor_controller
[WARN] [1670677771.406676121]: RRTConnect path optimization goals. Controller is not running.
[INFO] [1670677771.406793908]: Sent program to robot
[INFO] [1670677771.472716307]: Robot connected to reverse interface. Ready to receive control comman
ds.

[INFO] [1670677820.302392533]: manipulator/manipulator[RRTConnect]: Created 5 states (2 start + 3 go
als)
[INFO] [1670677820.302758063]: manipulator/manipulator[RRTConnect]: Created 5 states (3 start + 2 go
als)
[INFO] [1670677820.302972122]: manipulator/manipulator[RRTConnect]: Created 4 states (2 start + 2 go
als)
[INFO] [1670677820.303852617]: manipulator/manipulator[RRTConnect]: Created 5 states (3 start + 2 go
als)
[INFO] [1670677820.304028311]: ParallelPlan::solve(): Solution found by one or more threads in 0.002
405 seconds
[INFO] [1670677820.304243307]: manipulator/manipulator[RRTConnect]: Starting planning with 1 states
already in datastructure
[INFO] [1670677820.304399429]: manipulator/manipulator[RRTConnect]: Starting planning with 1 states
already in datastructure
[INFO] [1670677820.304738831]: manipulator/manipulator[RRTConnect]: Created 4 states (2 start + 2 go
als)
[INFO] [1670677820.305012186]: manipulator/manipulator[RRTConnect]: Created 4 states (2 start + 2 go
als)
[INFO] [1670677820.305178828]: ParallelPlan::solve(): Solution found by one or more threads in 0.001
935 seconds
[INFO] [1670677820.306739552]: SimpleSetup: Path simplification took 0.001499 seconds and changed fr
om 3 to 2 states
[INFO] [1670677822.361150636]: Execution request received
[INFO] [1670677826.213089655]: Controller 'scaled_pos_joint_traj_controller' successfully finished
[INFO] [1670677826.219075965]: Completed trajectory execution with status SUCCEEDED ...
[INFO] [1670677826.219218848]: Execution completed: SUCCEEDED

```

Figure 21 - ROS running sample

4. Results and Discussions

4.1. Yellow filter

The first test that was to be performed was to identify the main samples of the CRA lab (wooden blocks). The blocks are placed in the camera's normal view point and parameters are set in the Lab colour space as it isolated the yellow colour very well. The morphed area is converted to binary image. The 0's are filled by dilation and noise is removed by erosion followed by dilation. Figure 10 shows the only yellow filter and figure 11 shows extraction of final 2 objects only isolating them.

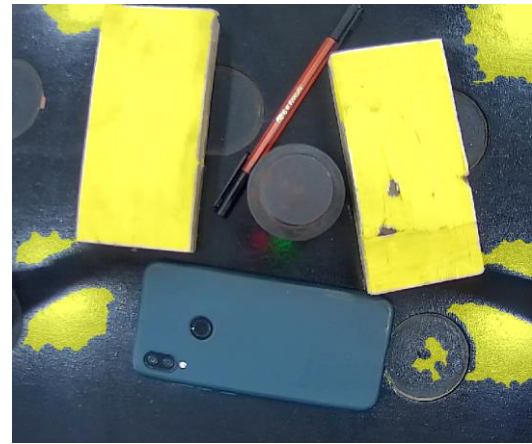


Figure 22 - colorization

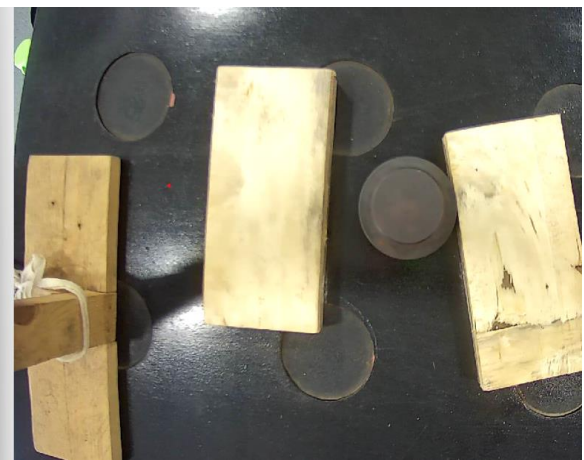


Figure 23- isolating the blocks

A bounding box is created around the detected objects. This function determines the location of all the sample blocks. In cases where 2 blocks are placed on top of each other or another object blocks some of the wooden block, it will still be highlighted and detected by the function. Some more experimental results are shown below.

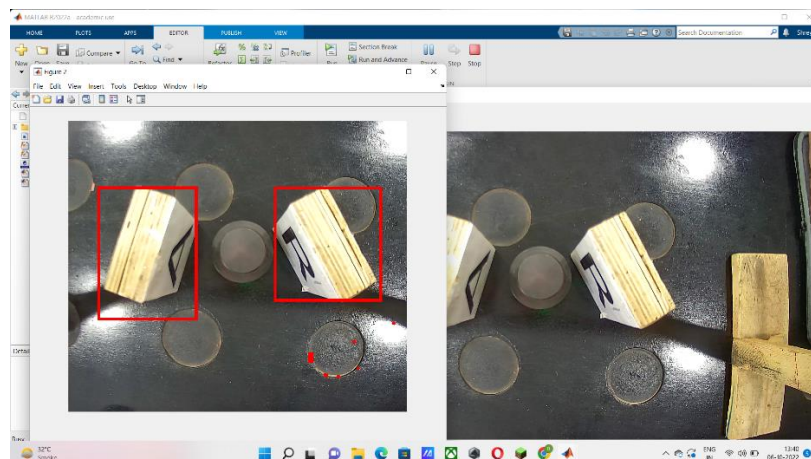


Figure 24 - Normal Placement

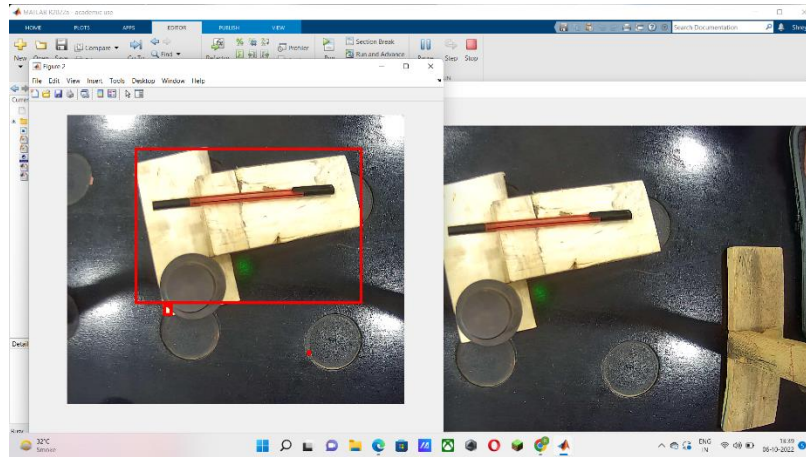


Figure 25 - Overlapping conditions

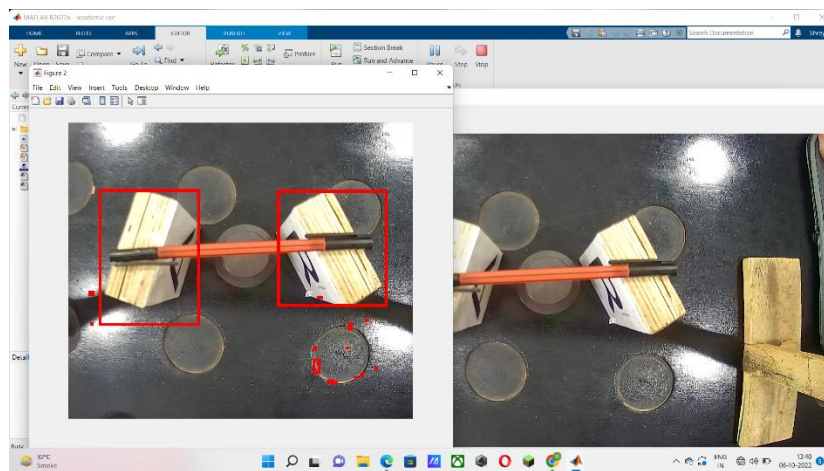


Figure 26 - Hanging condition

4.2. Reference filters for getting corner points

The coordinates of the objects are needed to perform IK or directly send them to the waypoints section of the teach pendant. Canny edge is used with blurred value of 10 and then circularity is used to get all the circles in the images. The real-time ratio and white colour filter are used to filter out unnecessary circles. After getting isolating the desired circles, the point-to-point distance and slope is calculated. The distance is then converted and kept as a constant for pixel to task space conversion. if the angle by reference points is different form real time values. It indicates a offset to be added every time to calculations. The results are shown below.

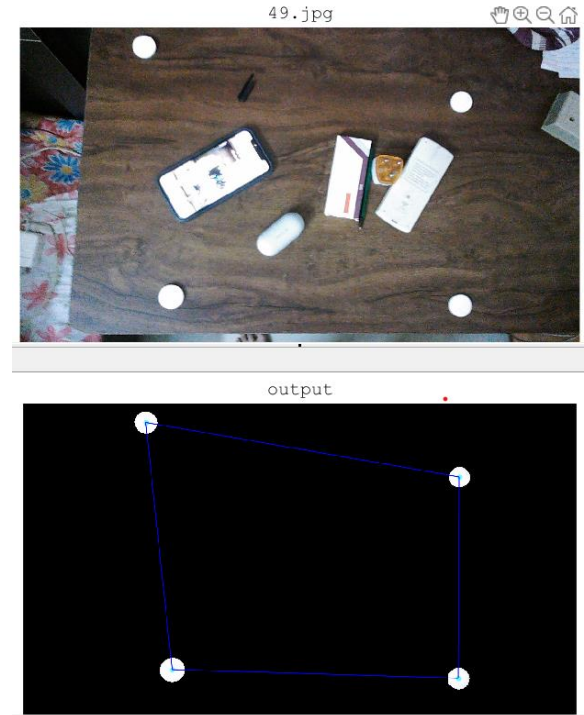
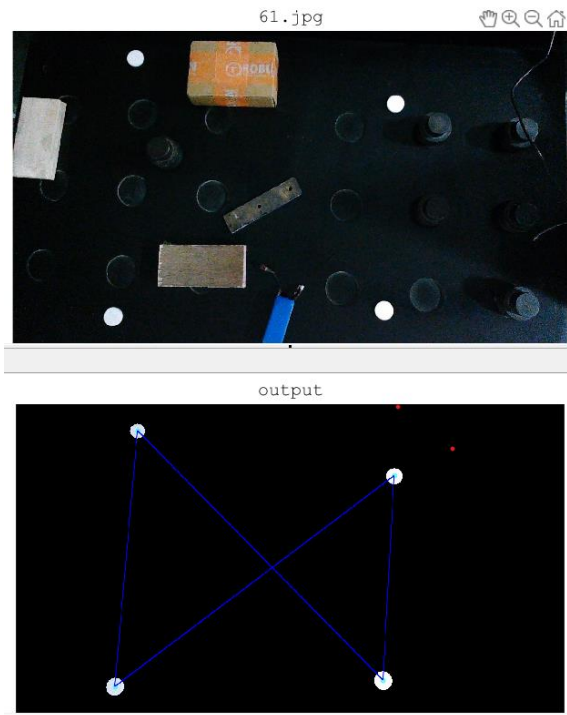


Figure 27 - Corner detection (a)

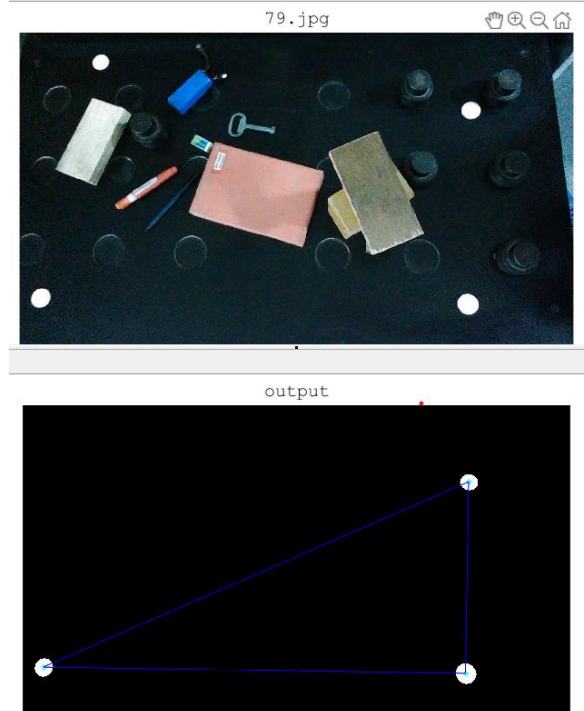
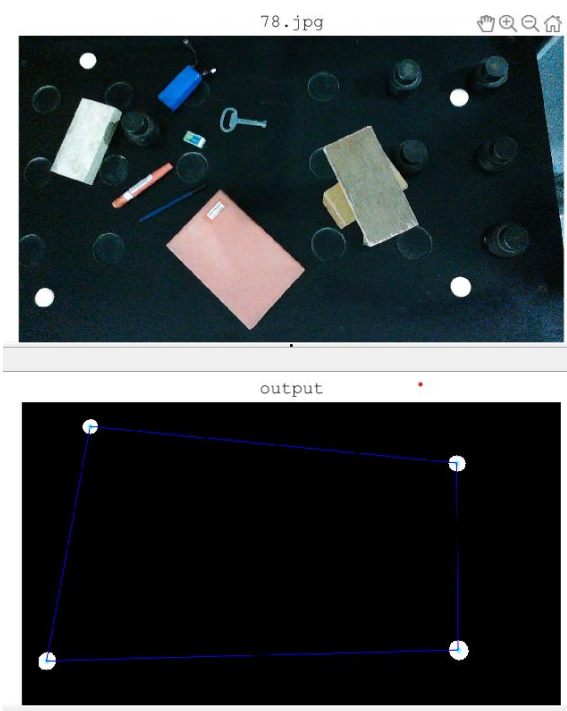


Figure 28 - Corner detection (b)

4.3. Detecting Rectangles and major minor axes

After canny edge detection and applying gaussian filter, some operations are performed to get rough true areas. then circularity is for detecting which figures are rectangles. Along with that the major axis and minor axis is found on the basis of filled area of true values in the binary

image. The axis of ellipse is then scaled and converted to fit the rectangle. The axes of rectangle also equate to the length and breadth of the object thus getting exact 2D view. The major axis's orientation represents the object as the gripper would prefer grasping object where it doesn't need to open more. The results of detecting rectangle. the results of rectangle detection are given below.

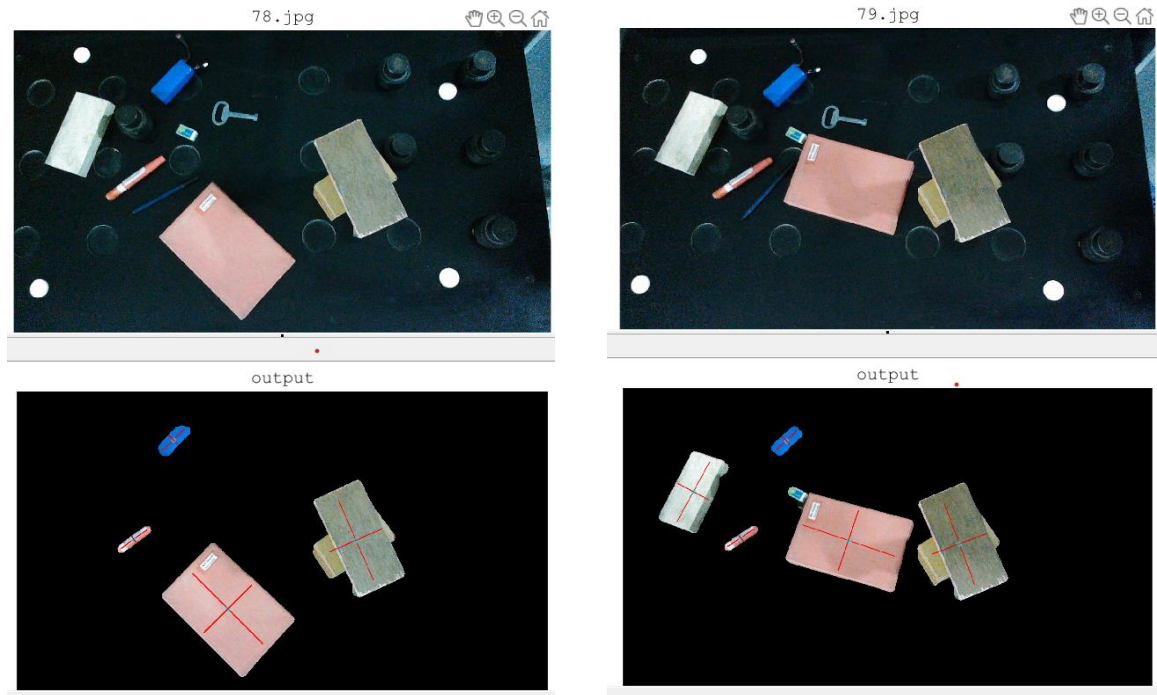


Figure 29 - Rectangle detection (a)



Figure 30 - Rectangle Detection (b)

The complete results integrating the distance from one of the chosen centre points to the detected objects is given below. The code chooses any of the points and adds the distance locating the detected objects centre by the pixel conversion ratio calculated before. Some of the results are given below.

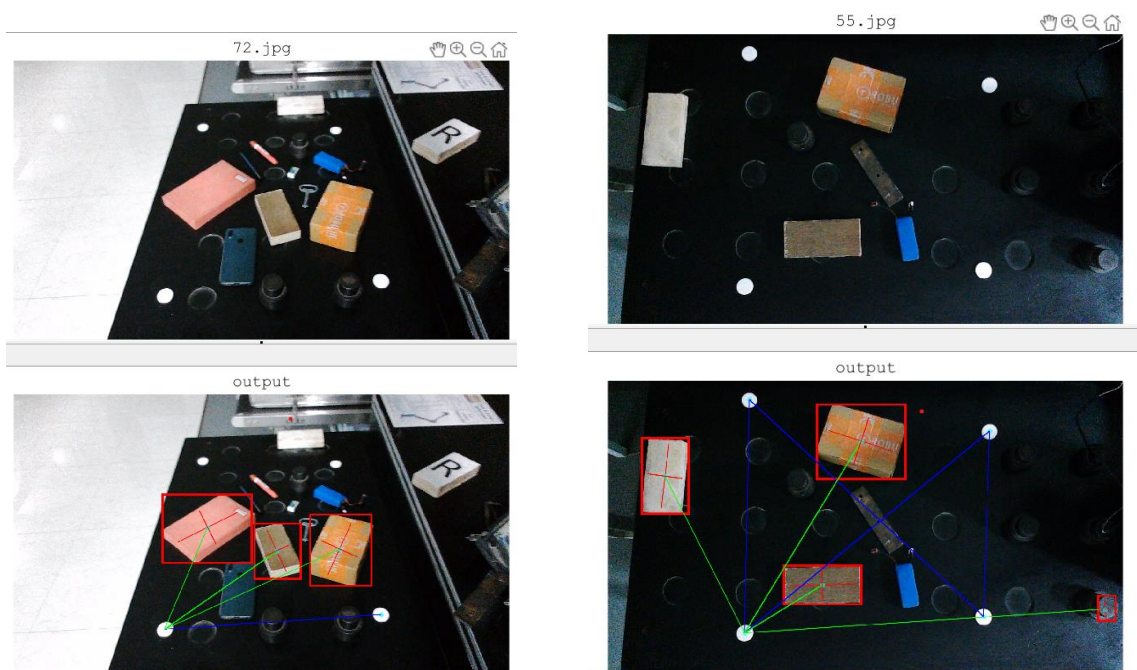


Figure 31- Complete results (a)

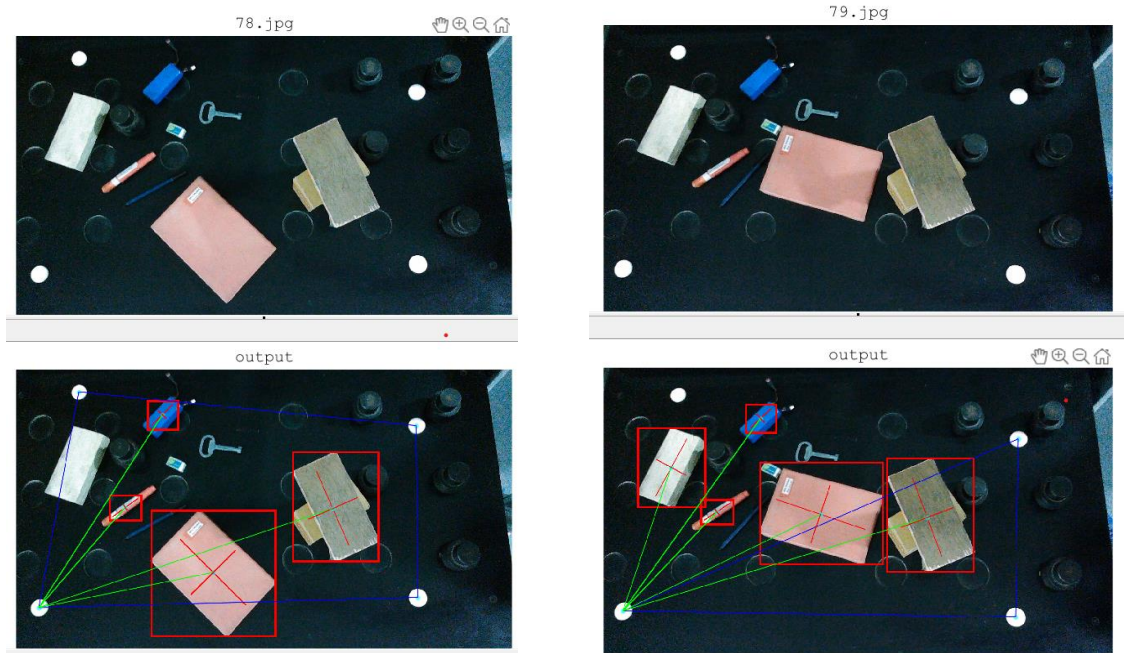


Figure 32 - Complete Results (b)

4.4. ROS and Rviz

As stated, before Robotic operating system is a special OS used for any and all robotic systems such as turtle bots, universal robots etc. The teach pendant needs to be configured for external control to use ROS. An ethernet cable is used for establishing a connection with the PC hosting the ROS. After setting up the necessary configurations, the robot needs to be calibrated first to get the current values. The values of all the joint states including power settings and program running in the teach pendant is extracted. The values and states are then used to start a rosmaster with universal robot 10e. the following snippets depict the successful calibration of the robot. The ubuntu operating system is necessary for running ros or virtual machine would work too.

```

/home/yuzaki/catkin_ws/src/universal_robot/ur10e_moveit_config/launch/moveit_planning_execution.launch http://192.168.43.123:11311
... logging to /home/yuzaki/.ros/log/3bf6234e-7a16-11ed-bd3f-496e92fc1ce7/roslaunch-yuzaki-ROG-Str
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.43.123:36021/

SUMMARY
=====
PARAMETERS
* /move_group/allow_trajectory_execution: True
* /move_group/capabilities:
* /move_group/controller_list: [{'name': 'scaled...
* /move_group/disable_capabilities:
* /move_group/jiggle_fraction: 0.05
* /move_group/planner_config: RRTConnect
* /move_group/planner_config/longest_valid_segment_fraction: 0.01
* /move_group/planner_config/planner_configs: 'SBL', 'EST', 'L...
* /move_group/max_range: 5.0
* /move_group/max_safe_path_cost: 1
* /move_group/moveit_controller_manager: moveit_simple_con...
* /move_group/moveit_manage_controllers: True
* /move_group/octomap_resolution: 0.025
* /move_group/planner_configs/BFRT/balanced: 0
* /move_group/planner_configs/BFRT/cache_ccc: 1
* /move_group/planner_configs/BFRT/extended_fmt: 1
* /move_group/planner_configs/BFRT/heuristic: 1
* /move_group/planner_configs/BFRT/nearest_k: 1
* /move_group/planner_configs/BFRT/num_samples: 1000
* /move_group/planner_configs/BFRT/optimal: 1
* /move_group/planner_configs/BFRT/robot_model: ur10e
* /move_group/planner_configs/BKPIECE/border_fraction: 0.9
* /move_group/planner_configs/BKPIECE/failed_expansion_score_factor: 0.5
* /move_group/planner_configs/BKPIECE/min_valid_path_fraction: 0.5
* /move_group/planner_configs/BKPIECE/range: 0.0
* /move_group/planner_configs/BKPIECE/type: geometric:BKPIECE
* /move_group/planner_configs/BLEST/range: 0.0
* /move_group/planner_configs/BLEST/type: geometric:BLEST
* /move_group/planner_configs/BLTTRT/cost_threshold: 1e300
* /move_group/planner_configs/BLTTRT/fraction: 0.1
* /move_group/planner_configs/BLTTRT/fraction_threshold: 0.0
* /move_group/planner_configs/BLTTRT/init_temperature: 100
* /move_group/planner_configs/BLTTRT/range: 0.0
* /move_group/planner_configs/BLTTRT/range_change_factor: 0.1
* /move_group/planner_configs/BLTTRT/type: geometric:BLTTRT
* /move_group/planner_configs/EST/goal_bias: 0.05
* /move_group/planner_configs/EST/range: 0.0
* /move_group/planner_configs/EST/type: geometric:EST

RViz

```

Figure 33 - Calibration testing

After Rosmaster and corresponding nodes have been launched, we need a software that controls the robot, here a external package called move it is installed and initiated. The matlab's universal robot toolbox also works the same but for simulation we can either use Gazebo, Rviz or URsim. Here moveit! and Rviz are interlinked hence using Rviz would be easier. The following snippets show the interfaces and initializing moveit! and Rviz.

```

/home/yuzaki/catkin_ws/src/universal_robot/ur10e_moveit_config/launch/moveit_planning_execution.launch http://192.168.43.123:11311
... logging to /home/yuzaki/.ros/log/3bf6234e-7a16-11ed-bd3f-496e92fc1ce7/roslaunch-yuzaki-ROG-Str
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://192.168.43.123:41877/

SUMMARY
=====
PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.15.15

NODES
/
rviz_rvizact_RoG_Str_G513RC_G513RC_7349_3572876755162093732 (rviz/rviz)

ROS_MASTER_URI=http://192.168.43.123:11311

process[rviz_rvizact_RoG_Str_G513RC_G513RC_7349_3572876755162093732-1]: started with pid [7363]
INFO [1678847321.904201008]: rviz version 1.14.19
INFO [1678847321.904274428]: compiled against Qt version 5.12.8
INFO [1678847321.904301252]: compiled against OGRE version 1.9.0 (Ghadamon)
INFO [1678847321.915275884]: Forcing OpenGL version 0
INFO [1678847322.106058408]: Stereo is NOT SUPPORTED
INFO [1678847322.106135781]: OpenGL device: AMD YELLOW_CARP (DRM 3.42.0, 5.15.0-50-generic, LLVM 12.0.0)
INFO [1678847322.106159127]: OpenGL version: 4.6 (GLSL 4.6) limited to GLSL 1.4 on Mesa system.
INFO [1678847325.414969388]: Loading robot model 'ur10e_robot'...
WARN [1678847325.44152254]: IK plugin for group 'manipulator' relies on deprecated API. Please in
element Initialize(robotModel, ...)
INFO [1678847325.44471547]: IK Using joint shoulder link -6.28319 6.28319
INFO [1678847325.44499865]: IK Using joint upper_arm link -6.28319 6.28319
INFO [1678847325.44454540]: IK Using joint forearm link -3.14159 3.14159
INFO [1678847325.44453587]: IK Using joint wrist_1 link -6.28319 6.28319
INFO [1678847325.44459426]: IK Using joint wrist_2 link -6.28319 6.28319
INFO [1678847325.44455992]: IK Using joint wrist_3 link -6.28319 6.28319
INFO [1678847325.44456301]: Looking in common namespaces for param name: manipulator/position_onl
y_ik
INFO [1678847325.445825177]: Looking in common namespaces for param name: manipulator/solve_type
INFO [1678847325.44769355]: Using solve type Distance
INFO [1678847325.506358040]: Starting planning scene monitor
INFO [1678847325.509918212]: Listening to /move_group/monitored_planning_scene'
INFO [1678847325.943624026]: Constructing new MoveGroup connection for group 'manipulator' in name
space ''
INFO [1678847327.072505674]: Ready to take commands for planning group manipulator.

RViz

```

Figure 34 - ROSlaunch with Moveit!

Rviz is a visualization software used for controlling the UR10 cobot or any other robot. Here one can set the scenes and add variables such as wind, gravity disturbance, inanimate objects, path and trajectory planning, scene geometry and much more. We will use the joint parameters in this project where the 6 joint parameters are varied according to inverse kinematics solution acquired from the matlab code. Rviz uses ros nodes for each task communicated via topics or logical decision. Snippets of Rviz are provided below for basic visualization and simulation.

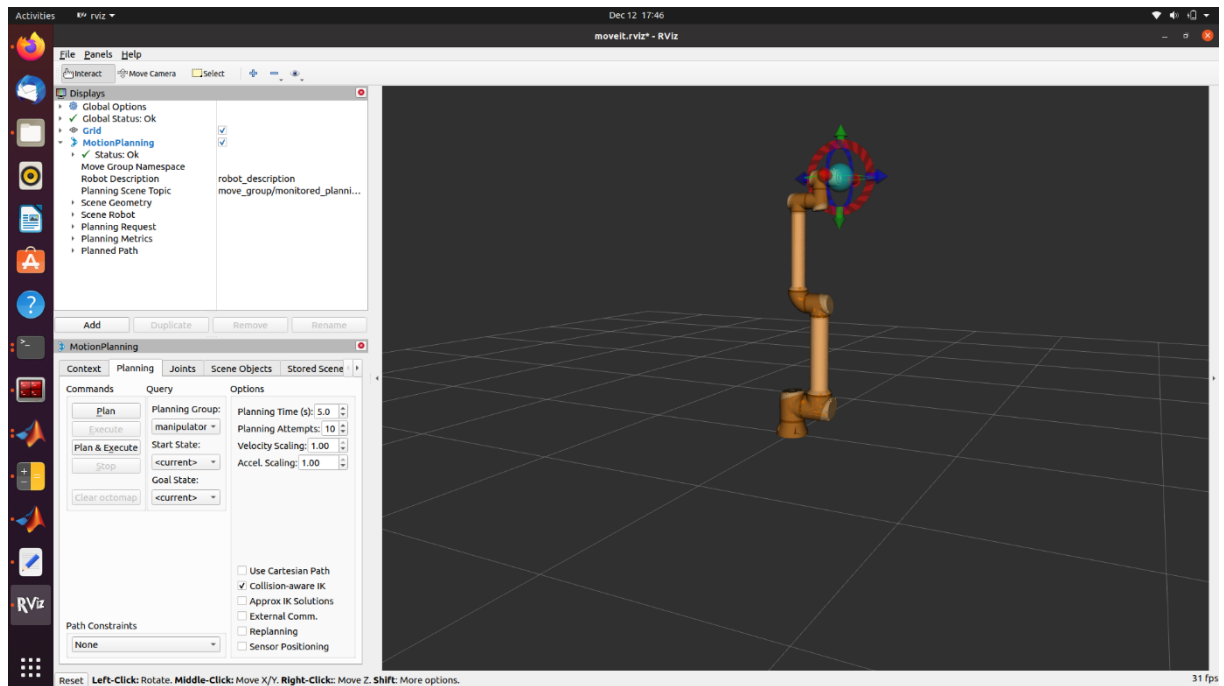


Figure 35 - Rviz demonstration

We can change the robot's TCP position using the drag and orient options at the tip of the robot. Or we can enter the TCP's cartesian coordinates using the x y z dx dy dz options which also includes the orientation. The joint angles calculated from the inverse kinematics are imported to the Rviz and the path planning is initiated. The path planning procedure of teach pendant takes precedence over the path described by the external software hence if an infeasible path or collision is about to happen which is missed in Rviz, teach pendant's software takes over and emergency stop is pulled. One example of combined working is given below.

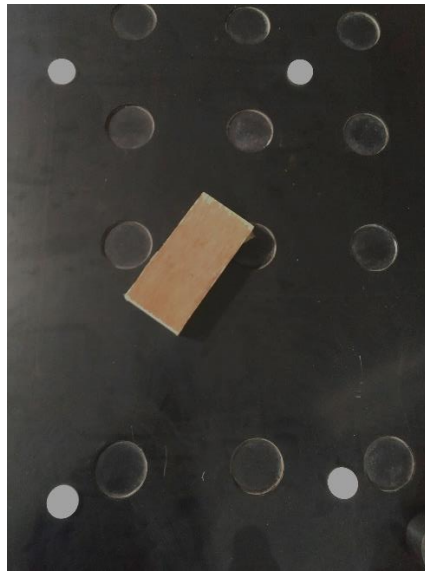


Figure 36 - Example with operating robot

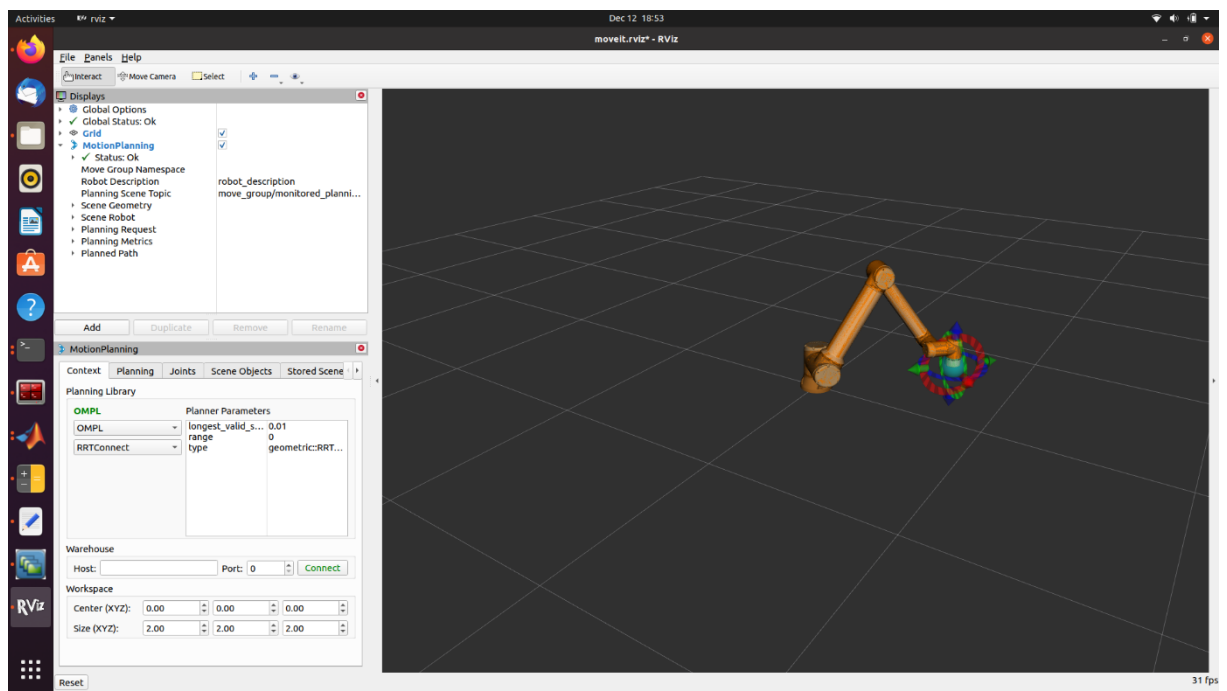


Figure 37 - Simulation of example in Rviz

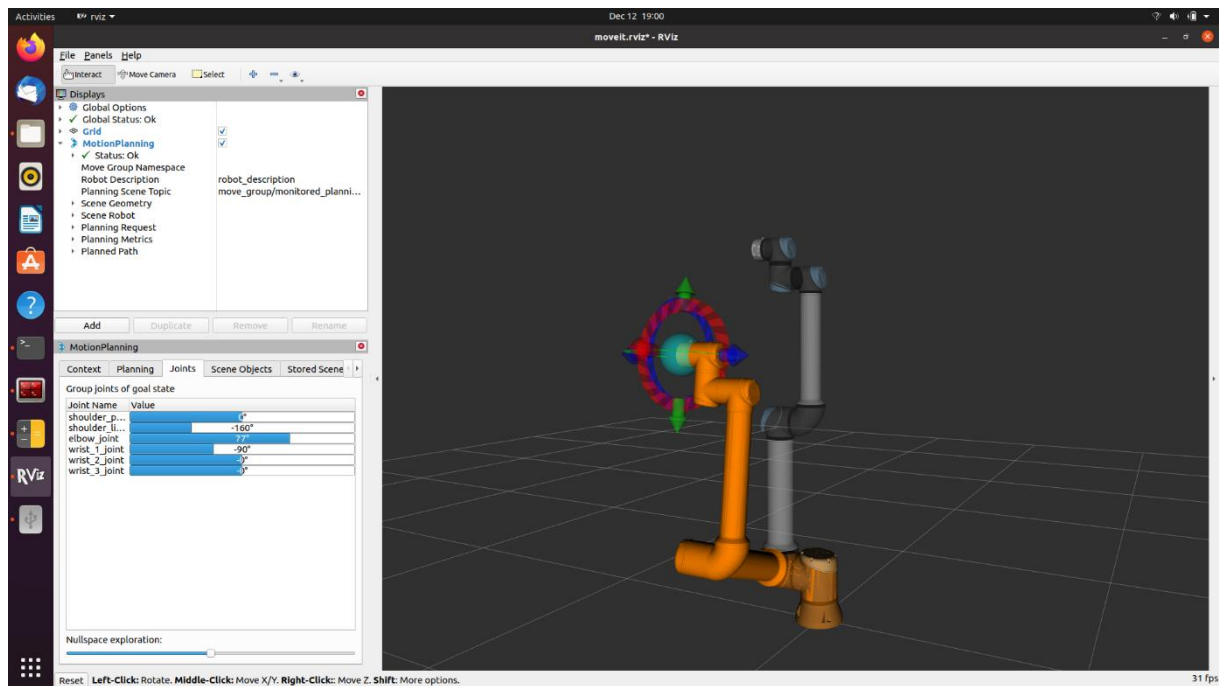


Figure 38 - Robot manoeuvring via moveit! and Rviz

From the above results we can clearly see that some noise is generated but the overall efficiency is quite high. Parameters like reflection of light proper configurations and modules like Matlab URcap file are a hinderance to the project. But otherwise, the results are satisfying for further development of the project. More professional tools like Gazebo can also be used for accurate simulation results.

5. Conclusion and Summary

Introduction to image processing and object detection was presented in the first chapter followed by various methods and simulation attempts made by researchers across the world is presented in the second chapter. The methodology and techniques used in this project are supplemented in the third chapter. It also highlights the important software and hardware as well as the dependencies needed in this project. The use of ROS and moveit! Is demonstrated which is used in the current study. The fourth chapter showcases the results of detecting specific objects along with simulation run and real time example.

It can be concluded that the current methodology can be used to get sufficiently good results in extracting rectangular shaped objects representing patients' medicines in clinics. However, there can be many improvements made to reduce noise in unexpected scenarios and making the robot adapt to severe conditions like high exposure to light or a blockage in path.

5.1. Future scope

The project could be further improved by introducing deep learning algorithms a step below the final step which would check if the extracted segments are really rectangles or not. Moreover, the project only focused on creating a 2D space for object recognition, stereo vision would be a better choice for reducing features like reflection blurred capture or missing geometric features. Using such a system for depth capture opens the door for more applications such as gesture recognition and complete automation.

6. References

- i. Wu, Juan & Peng, Bo & Huang, Zhenxiang & Xie, Jietao. (2013). Research on Computer Vision-Based Object Detection and Classification. IFIP Advances in Information and Communication Technology. 392. 183-188. 10.1007/978-3-642-36124-1_23.
https://www.researchgate.net/publication/287162496_Research_on_Computer_Vision-Based_Object_Detection_and_Classification
- ii. Singh, Kiran Jot & Kapoor, Divneet & Thakur, Khushal & Sharma, Anshul & Gao, Xiao-Zhi. (2022). Computer-Vision Based Object Detection and Recognition for Service Robot in Indoor Environment. Computers, Materials & Continua. 72. 197-213. 10.32604/cmc.2022.022989.
https://www.researchgate.net/publication/358823508_Computer-Vision_Based_Object_Detection_and_Recognition_for_Service_Robot_in_Indoor_Environment/citation/download
- iii. Kumar, Rahul & Kumar, Sanjesh & Lal, Sunil & Chand, Praneel. (2014). Object Detection and Recognition for a Pick and Place Robot. 10.13140/2.1.4379.2165.
https://www.researchgate.net/publication/271769133_Object_Detection_and_Recognition_for_a_Pick_and_Place_Robot/citation/download
- iv. Alejandra Cruz Bernal and Gilberto Moreno Aguilar. Article: Vision System via USB for Object Recognition and Manipulation with Scorbobot-ER 4U. *International Journal of Computer Applications* 56(18):10-15, October 2012.
<https://www.ijcaonline.org/archives/volume56/number18/8998-3174>
- v. Kumar, R., Kumar, S., Lal, S.P., & Chand, P. (2014). Object detection and recognition for a pick and place Robot. *Asia-Pacific World Congress on Computer Science and Engineering*, 1-7.
<https://www.semanticscholar.org/paper/Object-detection-and-recognition-for-a-pick-and-Kumar-Kumar/91cff13b948e683b38aa9dfbdde093713314659c>
- vi. Ali, M.H., & Mir-Nasiri, N. (2016). Vision Based Object Classification with Scorbobot-ER 4U.
<https://www.semanticscholar.org/paper/Vision-Based-Object-Classification-with-Scorbobot-ER-Ali-Mir-Nasiri/9e7612d5bf3a194c3af3dd771644a603feed7cc1>
- vii. Soh, Ji Yune & Lee, Junbok & Han, Choong-Hee. (2009). Development of Object Detection Technology Using Laser Sensor for Intelligent Excavation Work. 10.22260/ISARC2009/0034.
https://www.researchgate.net/publication/320560856_Development_of_Object_Detection_Technology_Using_Laser_Sensor_for_Intelligent_Excavation_Work/citation/download
- viii. <http://wiki.ros.org/noetic/Installation>
- ix. <https://in.mathworks.com/help/robotics/ug/load-predefined-robot-models.html>
- x. https://github.com/UniversalRobots/Universal_Robots_ROS_Driver
- xi. https://github.com/ros-industrial/universal_robot/tree/melodic-devel/ur10_moveit_config/launch
- xii. http://docs.ros.org/en/kinetic/api/moveit_tutorials/html/doc/quickstart_in_rviz/quickstart_in_rviz_tutorial.html
- xiii. https://github.com/lihuang3/ur5_ROS-Gazebo/tree/master/launch

Plagiarism Report

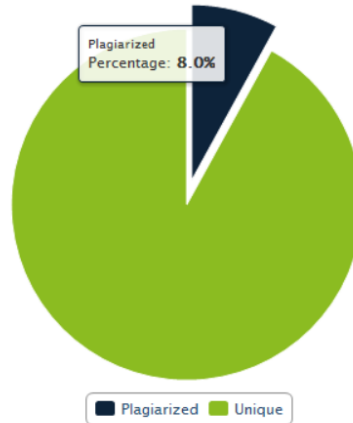
Summary Report

Print Save

Share Score with friends



PlagiarismCheckerX Summary Report



Detailed Report



Plagiarism Checker X Originality Report

Similarity Found: 8%

Date: Tuesday, December 13, 2022

Statistics: 477 words Plagiarized / 5855 Total words

Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.

Implementation of vision system on UR10e Cobot submitted in Partial Fulfilment of the Requirements for the Degree of BACHELOR OF TECHNOLOGY IN MECHANICAL ENGINEERING Submitted By: Shrey D Shah (19BME134) / DEPARTMENT OF MECHANICAL ENGINEERING, SCHOOL OF ENGINEERING, INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY, 2022

