**Department of Electrical Engineering**

**Faculty Member:** Hafsa Iqbal **Date:** April 16, 2024

**Semester:** 8  **Group:** 01

# EE381 Robotics

**Lab 9: Introduction to OpenCV**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO5-CLO4** | | **PLO5-CLO5** | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Viva / Quiz / Demo** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics** | **Individual and Team Work** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
| Hassan Rizwan | 335753 |  |  |  |  |  |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |  |  |
| Muhammad Umer | 345834 |  |  |  |  |  |
| Muhammad Abdullah Sohail | 343642 |  |  |  |  |  |
| Muhammad Saad | 333414 |  |  |  |  |  |

**Table of Contents**

**Lab 9: Introduction to CV3**

Introduction3

Objectives3

Lab Conduct3

Theory4

**Lab Task 1 – Load and Display Images**5

**Lab Task 2 – Cropping**6

**Lab Task 3 – Modifying Pixel Colors**7

**Lab Task 4 – Placing Shapes and texts**9

**Lab Task 5 – Resizing Images**10

**Lab Task 6 – Rotating Images**12

**Lab Task 7 – Visual Sensing in ROS**13

**Conclusion**14

# Lab 9: Introduction to OpenCV

## Introduction

This laboratory exercise will introduce OpenCV which is a popular and widely used library for image processing and computer vision applications. The field of computer vision is very important for robotics since it can be used to extract important information of a robot’s environment. A large number of algorithms used in robotics employ aspects of vision such as in feature extraction, image stitching, stereovision, visual servoing and structure from motion etc. Before starting computer vision, it is important to familiarize with the basics of image processing which is the subject of this lab.

## Objectives

* Load, save and display image data using Python
* Access and modify pixels as well as ROIs in images
* Place lines, rectangles, circles and text in images
* Resize image at various scales
* Rotate image at various angles

## Lab Conduct

* Respect faculty and peers through speech and actions
* The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
* In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
* Use the tab key to provide the indentation in python.

1. **Theory**

OpenCV is a library that focuses on image processing and computer vision. An image is an array of colored square called pixels. Each pixel has a certain location in the array and color values in BGR format. By referring to the array indices, the individual pixels or a range of pixels can be accessed and modified. OpenCV provides many functions for resizing, rotating, and placing objects in images. Rotation involves computing a 2-D rotation matrix which is applied for the transformation of the image.

Common terminal commands for ROS 2 are provided below:

**colcon build --packages-select <package\_name>**

build a specific package whenever a node is created or modified

**. install/setup.bash**

make the terminal “aware” of the workspace (notice the dot and space)

**ros2 pkg create --build-type ament\_python <package\_name>**

create a new package (must be done in src directory)

**ros2 pkg create --build-type ament\_python <package\_name> --dependencies rclpy std\_msgs geometry\_msgs sensor\_msgs**

create a new package with dependencies

**ros2 run <package\_name> <node\_name>**

execute a node in the terminal

In the lab, you will need to download a number of image files which will be used in the tasks.

**Lab Task 1 – Load and Display Images**

Write a python script in which you will load 3 images from disk. Then, display the images in different windows at the same time. You will need to provide the code and a single screenshot which shows all 3 windows.

***### TASK 1 CODE STARTS HERE ###***

"""

Write a python script in which you will load 3 images from disk. Then, display

the images in different windows at the same time. You will need to provide the

code and a single screenshot which shows all 3 windows.

"""

import cv2

img1 = cv2.imread("./stanford.png")

img2 = cv2.imread("./mit.png")

img3 = cv2.imread("./ucb.png")

cv2.imshow("img1", img1)

cv2.imshow("img2", img2)

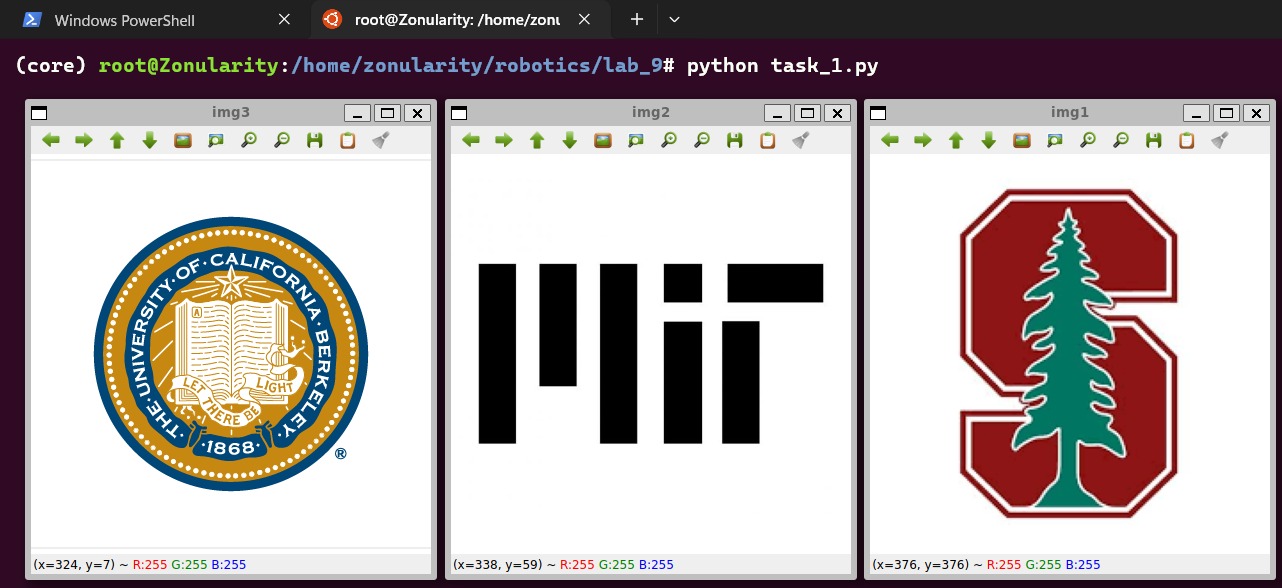
cv2.imshow("img3", img3)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 1 CODE ENDS HERE ###***

***### TASK 1 SCREENSHOT STARTS HERE ###***



***### TASK 1 SCREENSHOT ENDS HERE ###***

**Lab Task 2 – Cropping**

Write code to load an image file. Using the slice operation, crop out the four quadrants of the image and display them in separate windows. The code must be generic enough to take the image size into account. For submission, provide the code and a single screenshot showing all 4 windows.

***### TASK 2 CODE STARTS HERE ###***

"""

Write code to load an image file. Using the slice operation, crop out the four

quadrants of the image and display them in separate windows. The code must be

generic enough to take the image size into account. For submission, provide the

code and a single screenshot showing all 4 windows.

"""

import cv2

img = cv2.imread("./stanford.png")

height, width = img.shape[:2]

top\_left = img[: height // 2, : width // 2]

top\_right = img[: height // 2, width // 2 :]

bottom\_left = img[height // 2 :, : width // 2]

bottom\_right = img[height // 2 :, width // 2 :]

cv2.imshow("top\_left", top\_left)

cv2.imshow("top\_right", top\_right)

cv2.imshow("bottom\_left", bottom\_left)

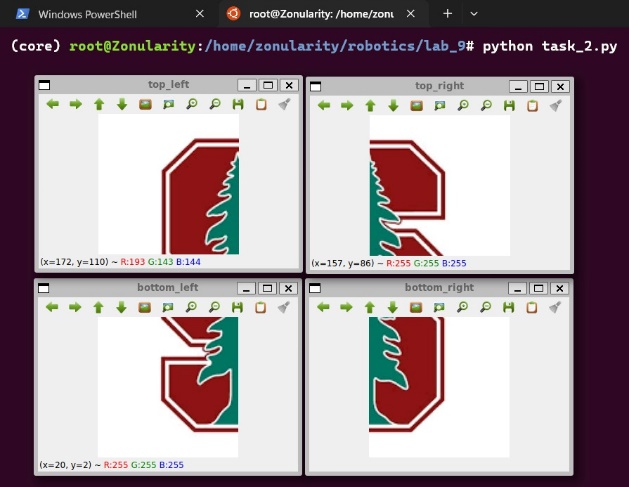
cv2.imshow("bottom\_right", bottom\_right)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 2 CODE ENDS HERE ###***

***### TASK 2 SCREENSHOT STARTS HERE ###***



***### TASK 2 SCREENSHOT ENDS HERE ###***

1. **Lab Task 3 – Modifying Pixel Colors**

Write code to load the image files and place alternating green and white horizontal lines in the image. Do **NOT** use the line function (cv2.line). You need to do this by changing the pixel colors. Each line is 1-pixel thick. The lines are also spaced apart by 1-pixel wide gap. Thus, the image will have one green line, then one line of image pixels, then one white line, then another line of image pixels and so on. Provide the code and screenshot for the submission

***### TASK 3 CODE STARTS HERE ###***

"""

Write code to load the image files and place alternating green and white

horizontal lines in the image. Do NOT use the line function (cv2.line). You need

to do this by changing the pixel colors. Each line is 1-pixel thick. The lines

are also spaced apart by 1-pixel wide gap. Thus, the image will have one green

line, then one line of image pixels, then one white line, then another line of

image pixels and so on. Provide the code and screenshot for the submission

"""

import cv2

img = cv2.imread("./stanford.png")

for i in range(0, img.shape[0], 2):

    if i + 2 <= img.shape[0] - 1:

        img[i] = [0, 255, 0]

        img[i + 2] = [255, 255, 255]

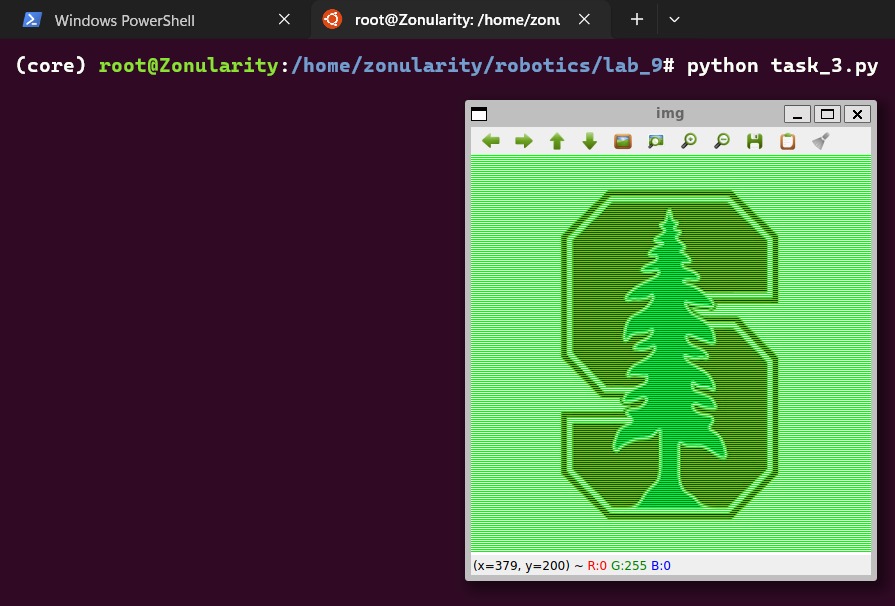
cv2.imshow("img", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 3 CODE ENDS HERE ###***

***### TASK 3 SCREENSHOT STARTS HERE ###***



***### TASK 3 SCREENSHOT ENDS HERE ###***

**Lab Task 4 – Placing Shapes and Text**

Load any one of the images and place a line, rectangle, circle and text using the inbuilt functions in OpenCV. Each of the placed object must have a different color. The text must contain the names of members of your group. Provide the code and screenshot of the image.

***### TASK 4 CODE STARTS HERE ###***

"""

Load any one of the images and place a line, rectangle, circle and text using

the inbuilt functions in OpenCV. Each of the placed object must have a different

color. The text must contain the names of members of your group. Provide the

code and screenshot of the image.

"""

import cv2

img = cv2.imread("./stanford.png")

height, width, channels = img.shape

img = cv2.line(img, (0, 0), (width, height), (255, 0, 0), 5)

img = cv2.rectangle(img, (30, 30), (width // 4, height // 4), (0, 255, 0), 5)

img = cv2.circle(

    img, (int(width / 3), int(height / 3)), int(height / 3), (0, 0, 255), 5

)

img = cv2.putText(

    img,

    "Ahmed Mohsin",

    (10, int(height) - int(height / 50)),

    cv2.FONT\_HERSHEY\_SIMPLEX,

    1,

    (0, 0, 0),

    2,

)

cv2.imshow("img", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 4 CODE ENDS HERE ###***

***### TASK 4 SCREENSHOT STARTS HERE ###***



***### TASK 4 SCREENSHOT ENDS HERE ###***

**Lab Task 5 – Resizing Images**

Load any one of the images and use the resize function to make copies of the image at different sizes. Display at least 3 different sizes in separate windows and take the screenshot. Provide the code and screenshot for the submission.

***### TASK 5 CODE STARTS HERE ###***

"""

Load any one of the images and use the resize function to make copies of the

image at different sizes. Display at least 3 different sizes in separate windows

and take the screenshot. Provide the code and screenshot for the submission.

"""

import cv2

img = cv2.imread("./stanford.png")

height, width, channels = img.shape

img\_half = cv2.resize(img, (width // 2, height // 2))

img\_double = cv2.resize(img, (width \* 2, height \* 2))

img\_quarter = cv2.resize(img, (width // 4, height // 4))

cv2.imshow("img\_half", img\_half)

cv2.imshow("img\_double", img\_double)

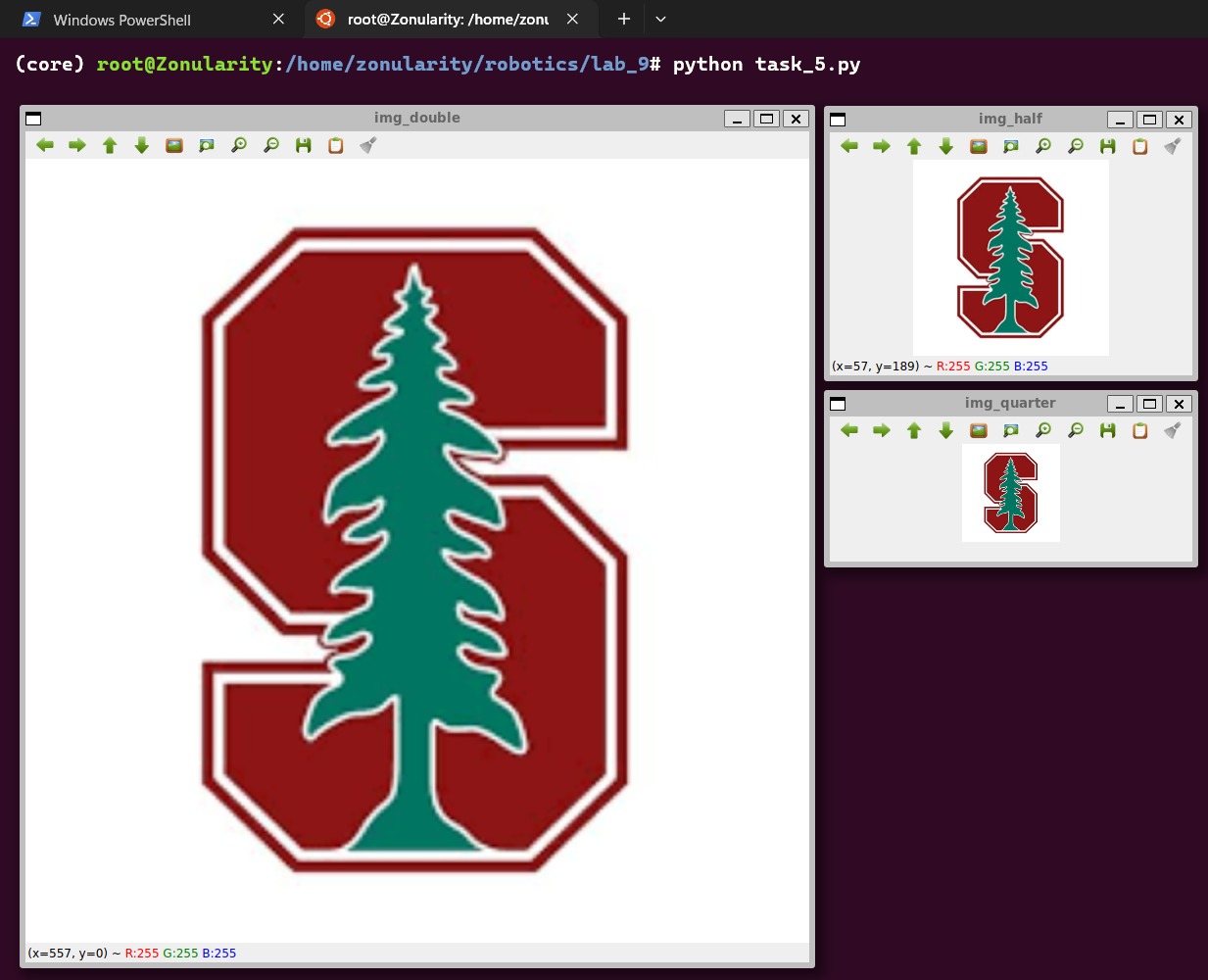
cv2.imshow("img\_quarter", img\_quarter)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 5 CODE ENDS HERE ###***

***### TASK 5 SCREENSHOT STARTS HERE ###***



***### TASK 5 SCREENSHOT ENDS HERE ###***

**Lab Task 6 – Rotating Images**

Load any one of the images and use the rotate function to rotate the image at angles of 30, 60 and 90 degrees. For each rotated image, you need to manually adjust the scale factor (in get2DRotationMatrix function) so that the entire image is shown in the window. The rotated image’s border/corner must touch the window’s border. Show all 3 windows in the screenshot. Provide the code and screenshot for the submission.

***### TASK 6 CODE STARTS HERE ###***

"""

Load any one of the images and use the rotate function to rotate the image at

angles of 30, 60 and 90 degrees. For each rotated image, you need to manually

adjust the scale factor (in get2DRotationMatrix function) so that the entire

image is shown in the window. The rotated image's border/corner must touch the

window's border. Show all 3 windows in the screenshot. Provide the code and

screenshot for the submission.

"""

import cv2

img = cv2.imread("./stanford.png")

height, width, channels = img.shape

img\_30 = cv2.getRotationMatrix2D((width / 2, height / 2), 30, 0.73)

img\_30 = cv2.warpAffine(img, img\_30, (width, height))

img\_60 = cv2.getRotationMatrix2D((width / 2, height / 2), 60, 0.73)

img\_60 = cv2.warpAffine(img, img\_60, (width, height))

img\_90 = cv2.getRotationMatrix2D((width / 2, height / 2), 90, 1)

img\_90 = cv2.warpAffine(img, img\_90, (width, height))

cv2.imshow("img\_30", img\_30)

cv2.imshow("img\_60", img\_60)

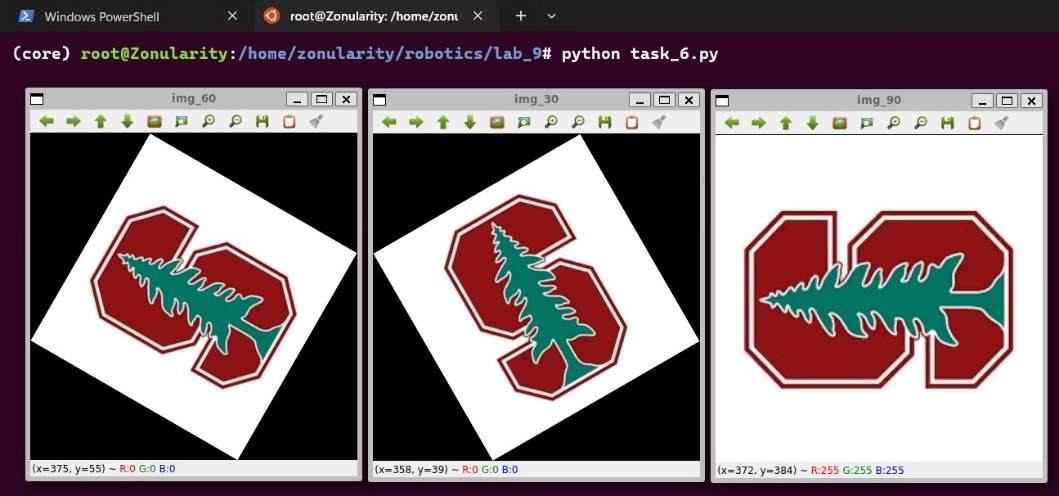
cv2.imshow("img\_90", img\_90)

cv2.waitKey(0)

cv2.destroyAllWindows()

***### TASK 6 CODE ENDS HERE ###***

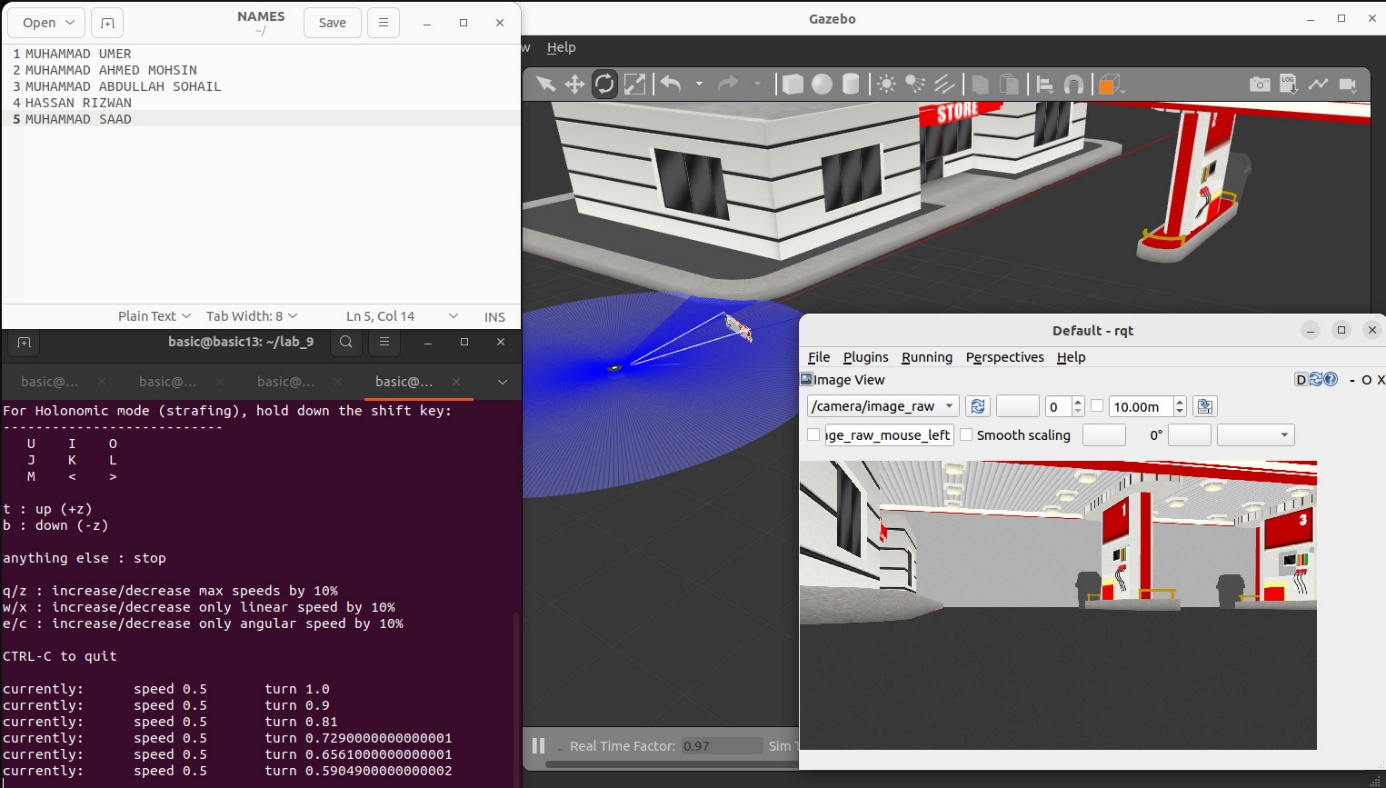
***### TASK 6 SCREENSHOT STARTS HERE ###***



***### TASK 6 SCREENSHOT ENDS HERE ###***

**Lab Task 7 – Visual Sensing in ROS**

Load the ROS simulation with a robot that has a camera sensor mounted on it. Use the RQt (Visualization -> Image View) to display the output of the visual sensing. Place a few objects in the environment within the field of view. For the task submission, provide a screenshot of the simulation which shows both the robot and the camera output.

***### TASK 7 SCREENSHOT STARTS HERE ###***

***### TASK 7 SCREENSHOT ENDS HERE ###***

**Conclusion:**

The tasks undertaken in this lab report provided valuable insights into the practical applications of computer vision within the realm of robotics. Furthermore, the integration of these skills with ROS, a widely used robotics middleware, showcased the importance of interoperability and standardization within the robotics community. This hands-on experience not only expanded our technical repertoire but also underscored the collaborative nature of robotics research and development. By leveraging the power of computer vision, we are poised to contribute to the advancement of robotic systems.