ROBOTICS LAB 11

Line Follower using Visual Feedback in ROS

MUNADI SIAL

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

Overview

This lab will be centered on the following:

- Use CV Bridge to convert image data to an OpenCV object
- Acquire masks for line following
- Determine centroid from the region of interest
- Use the shifting centroid to influence robot's motion

Introduction

So far we have looked at various concepts of ROS:

- Publishing to topics
- Subscribing from topics
- Twist Messages
- Laser Scanning

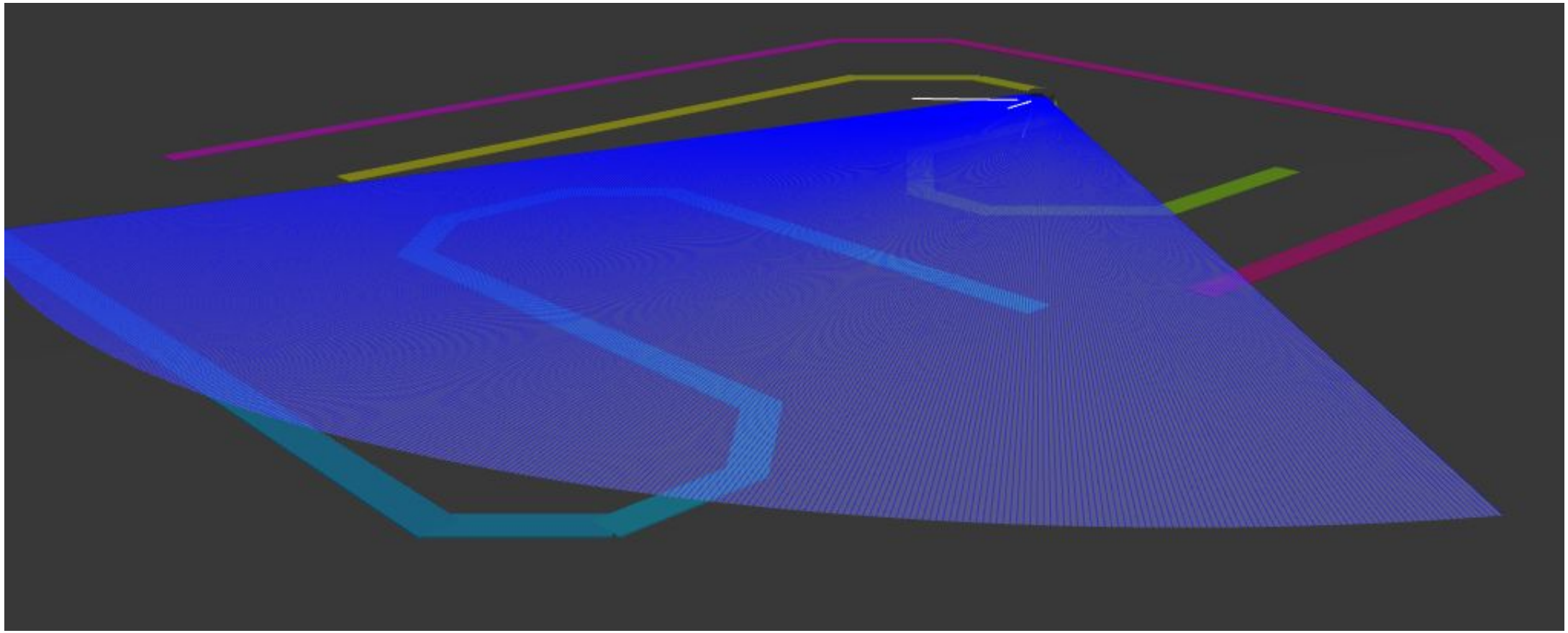
We have also looked at various aspects of OpenCV:

- Loading and displaying images
- Manipulating pixels and regions
- BGR to HSV color space conversion
- Masking using range of colors
- Centroid points

In this lab, we combine both ROS and OpenCV for a line following robot

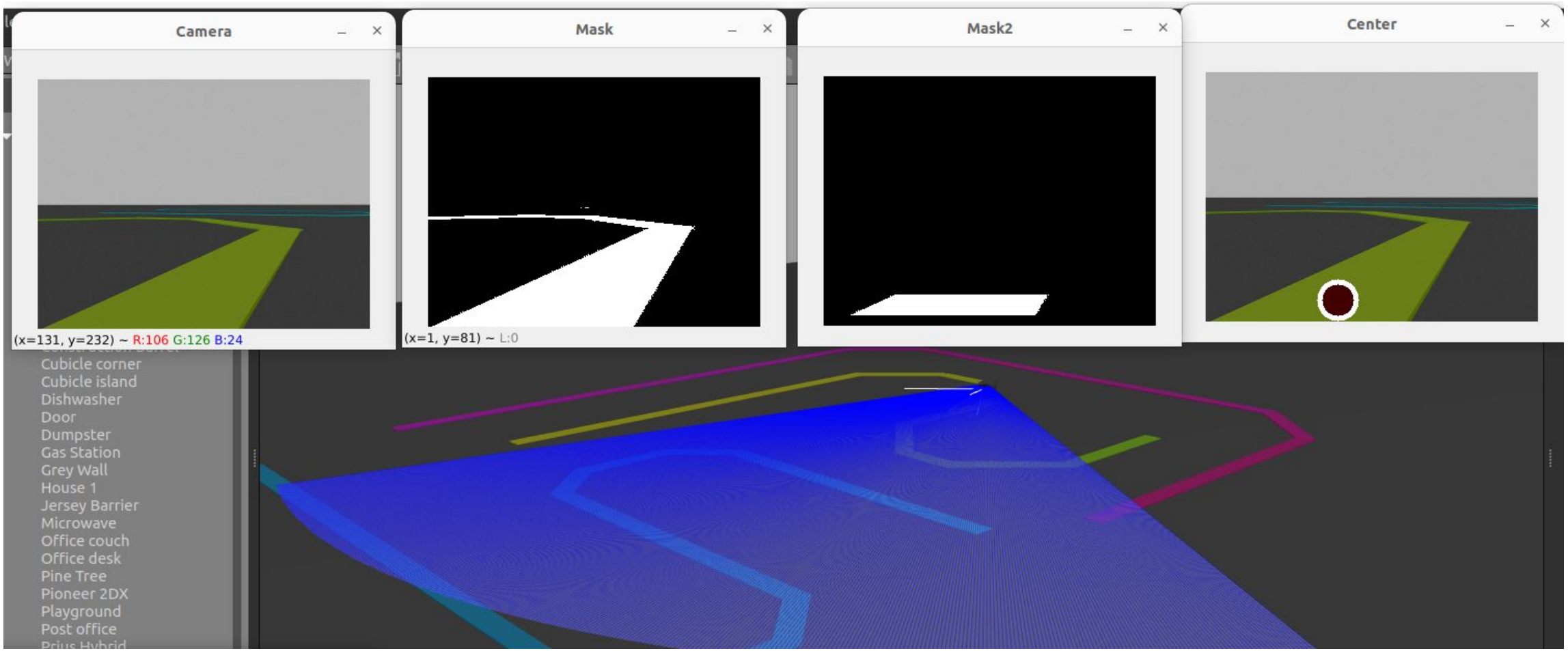
Line Following in ROS

- In this lab, you will be provided with a “lines” model for the track which can be placed in the simulation



Line Following in ROS

- You will use the robot's camera to get the image data and use OpenCV on the acquired images to make the line follower



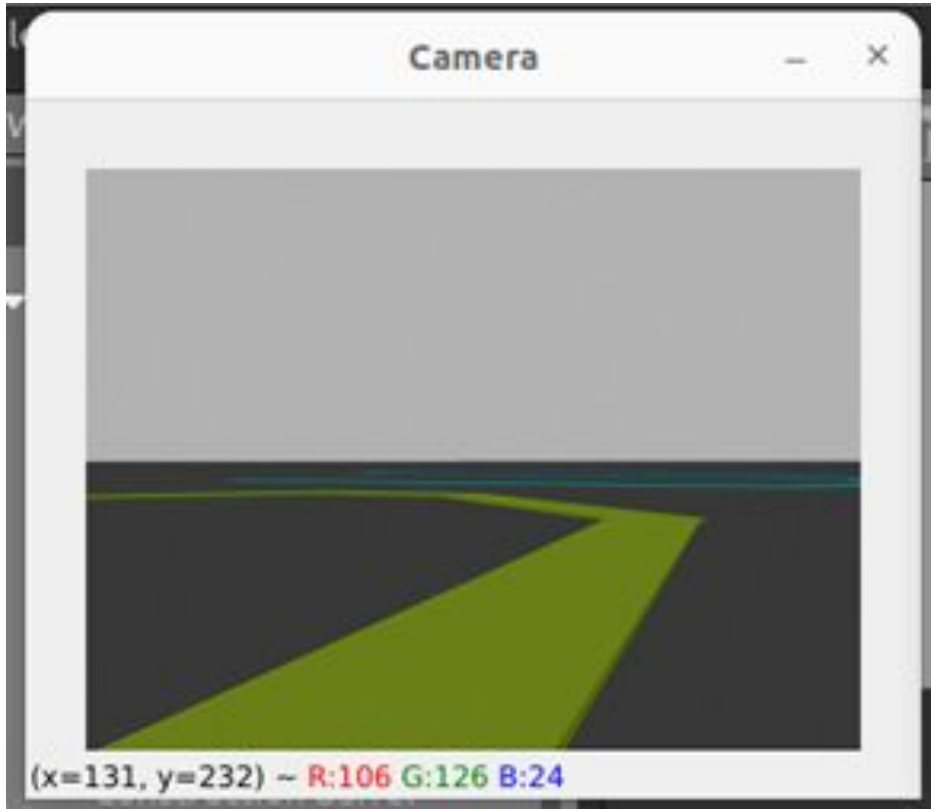
CV Bridge

The camera on the robot acquires the images

The images are sent to a **‘camera_color_frame/image_raw’** topic

A node can subscribe to the above topic to get the image data

The image data must be converted using the **CV Bridge** to an OpenCV object in order to use OpenCV functions in the node



CV Bridge

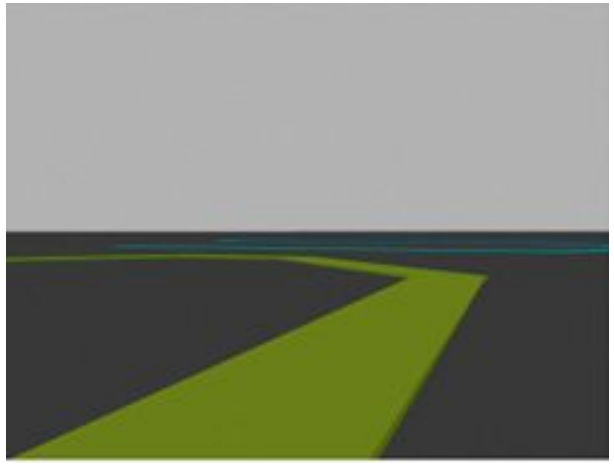
A simple CV Bridge use in a node is shown to display the images

```
class ImageSubscriber(Node):
  def __init__(self):
    super().__init__('image_subscriber')
    self.subscription = self.create_subscription(Image,'camera_color_frame/image_raw',
                                                 self.listener_callback,10)
    self.subscription
    self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
    self.move = Twist()
    self.br = CvBridge() # Used to convert between ROS and OpenCV images

  def listener_callback(self, data):
    self.get_logger().info('Receiving video frame')
    img = self.br.imgmsg_to_cv2(data, 'bgr8') # Convert ROS image to OpenCV image
    img = cv2.resize(img, None, 1, 0.5, 0.5, cv2.INTER_CUBIC)
    cv2.imshow('Center', img)
    cv2.waitKey(2)
```

Once, the images are obtained, the line follower program can be made

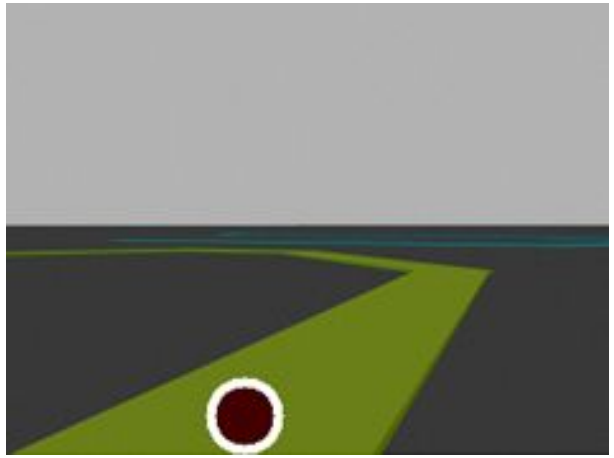
Line Follower with Vision



Camera

Threshold

Region of Interest

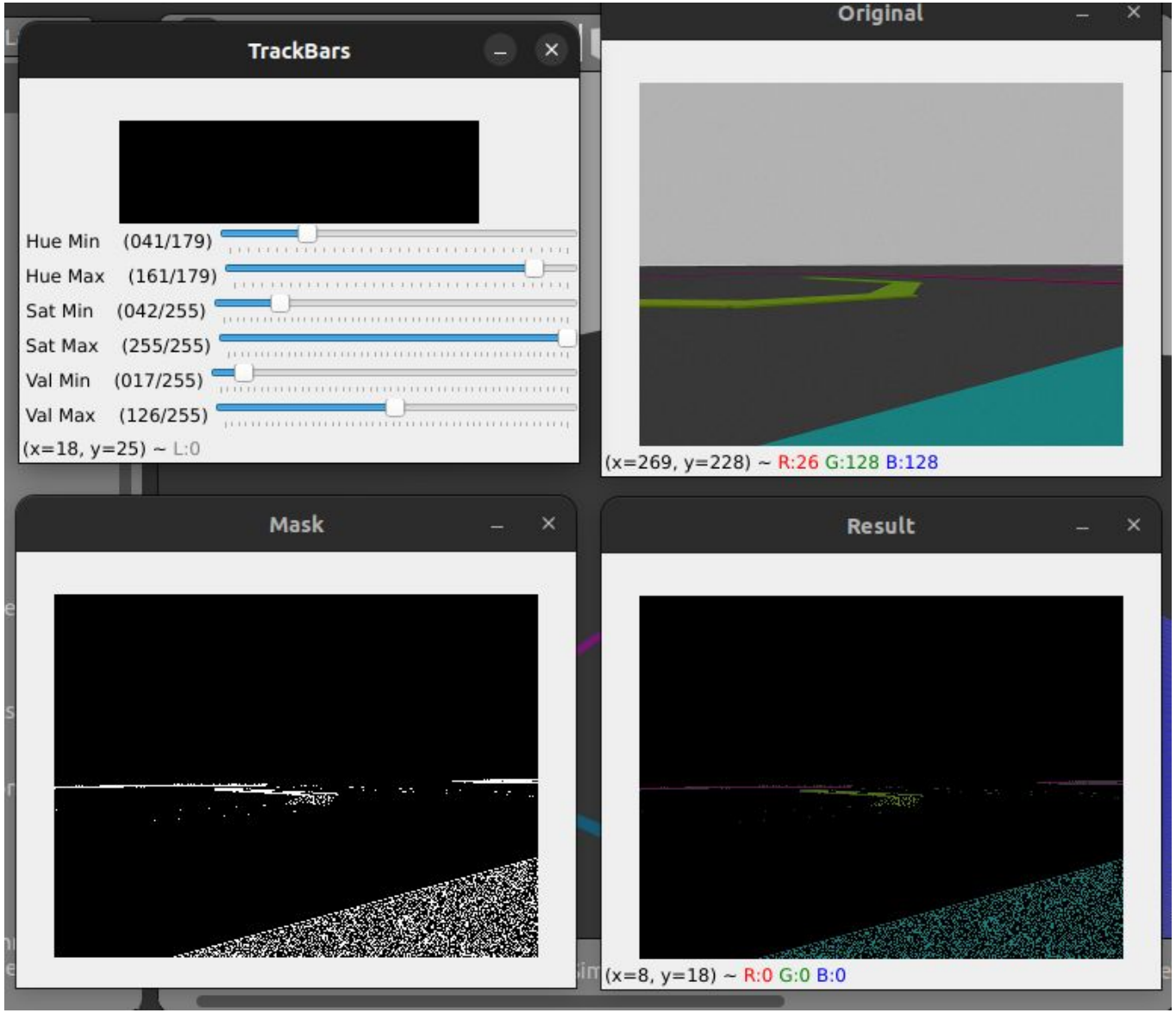


Centroid Indicator

Line Thresholding

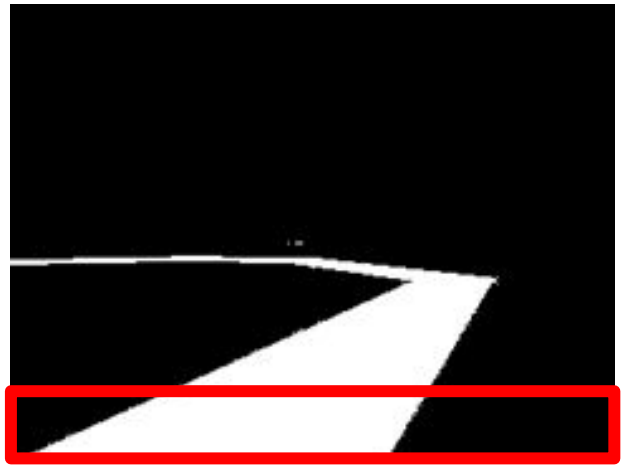
To mask the line properly, the HSV color space must be used

A Color Tracker program has been provided to get the range of H,S,V values for the threshold



Region of Interest

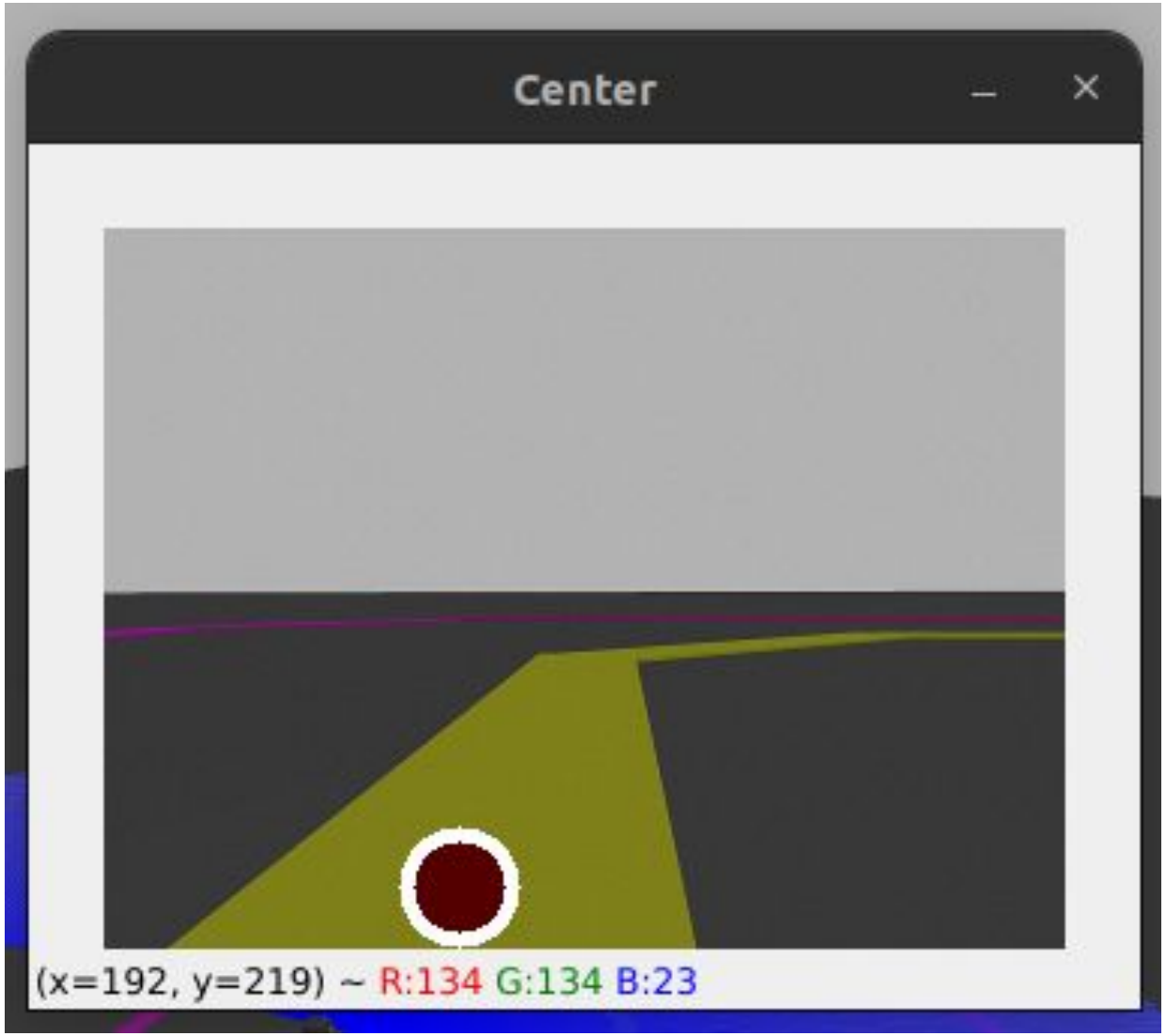
The bottom region in the threshold is obtained as we are only interest in the part of the line directly in front of the robot



Centroid Indicator

Once the region of interest is obtained, its centroid is computed to determine the center of the line

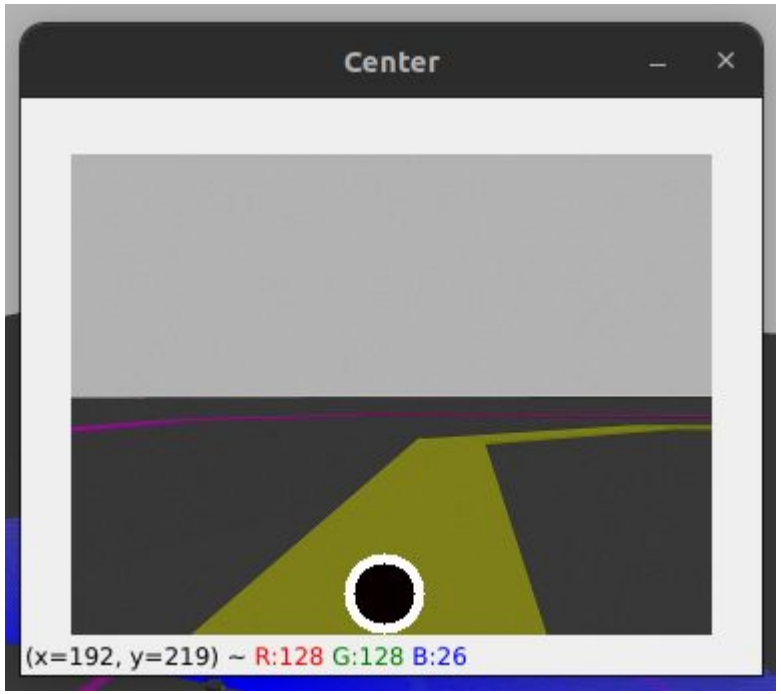
A circle is then drawn at the centroid point

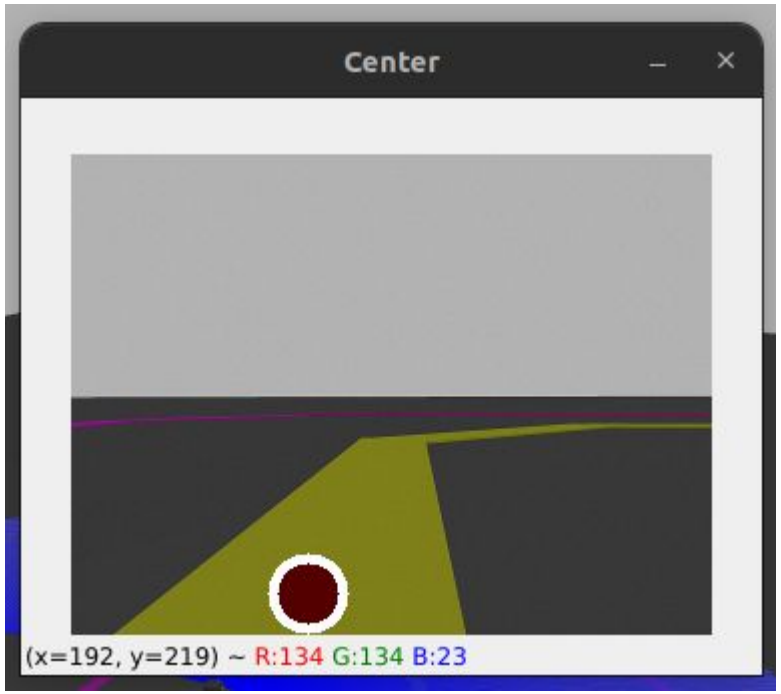


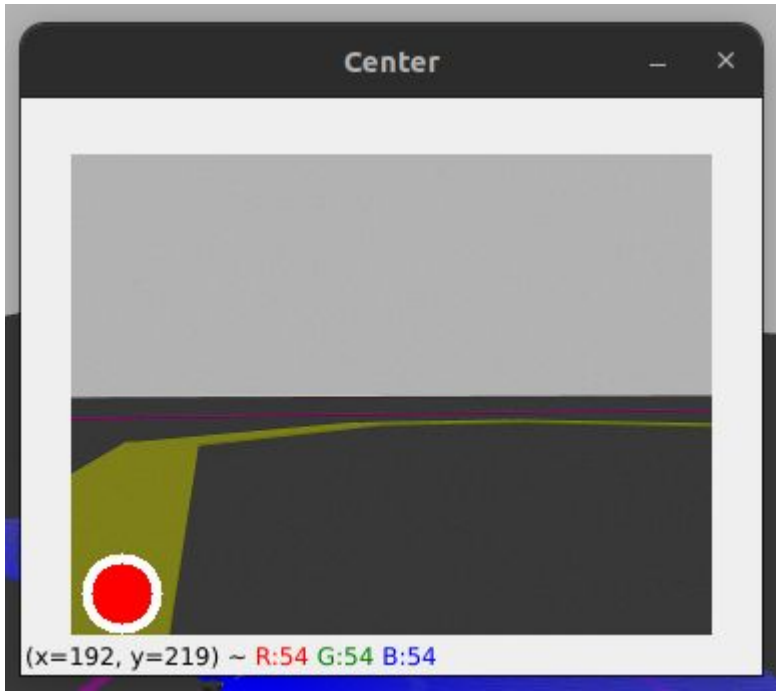
Centroid Indicator

The shift in the centroid along the horizontal will indicate whether the robot should turn right or left

The centroid circle changes color as it shifts more from the center







Motion Control

The difference between the centroid point and the image center point is used to compute the error

This error can be used to influence the robot's angular velocities

The magnitude of the angular velocity can be made proportional to the error

The robot must also have a linear velocity to keep moving on the line

The velocity values are published to the cmd_vel topic to actuate the line follower

Lab Tasks

- Download the manual from LMS
- Perform the Lab Tasks as given in the manual and submit it on LMS
- Remember to execute scripts with the terminal