Lab 6 – Report

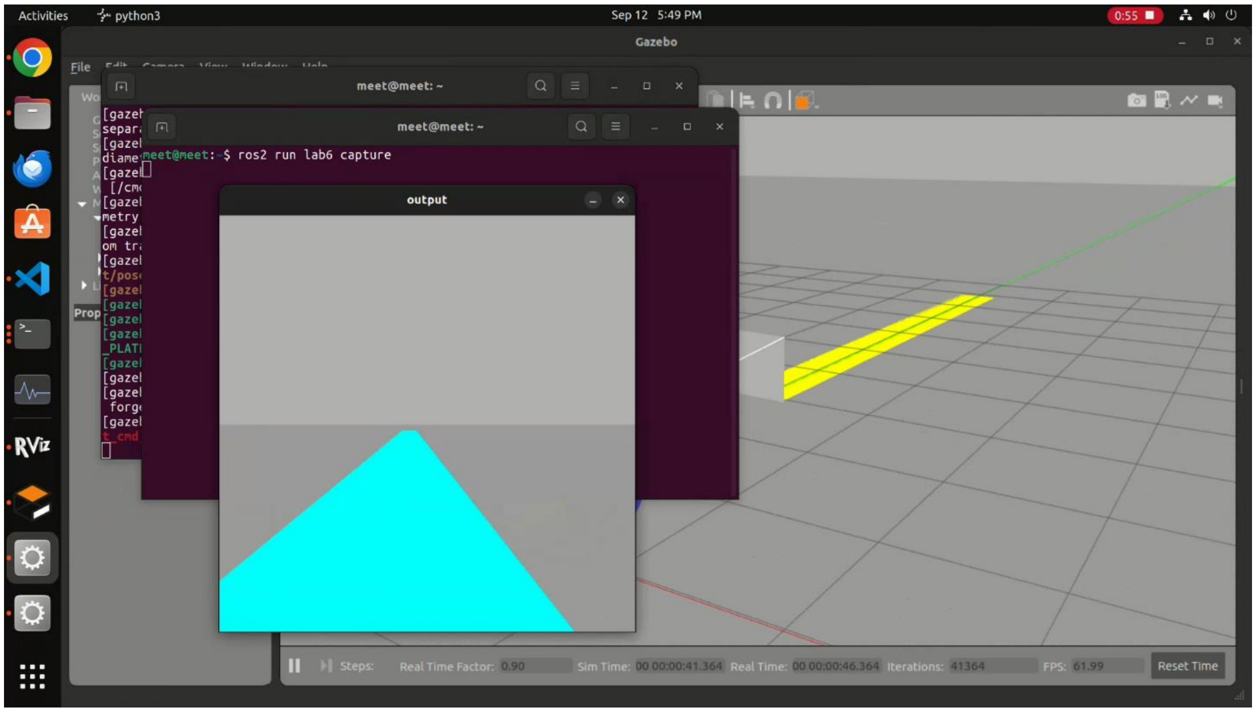
Meet Kansara – 220929270 Roll no. 54

Aim: To simulate a line-following robot in Gazebo that uses OpenCV to detect and follow a yellow line.

Code Execution and analysis:

1. Capturing image:

```
import rclpy
import cv2
from rclpy.node import Node
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
class Capture(Node):
  def __init__(self):
    super().__init__('video_subscriber')
    self.subscriber = self.create_subscription(Image,'/camera1/image_raw',self.process_data,10)
    self.out = cv2.VideoWriter('/home/meet/output.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10,
(512,512))
    self.bridge = CvBridge()
  def process_data(self, data):
    frame = self.bridge.imgmsg_to_cv2(data)
    self.out.write(frame)
    self.img = cv2.imwrite('/home/meet/shot.png', frame)
    cv2.imshow("output", frame)
    cv2.waitKey()
    cv2.destroyAllWindows()
def main(args=None):
  rclpy.init(args=args)
  node = Capture()
  rclpy.spin(node)
  rclpy.shutdown()
if __name__ == '__main__':
  main()
```



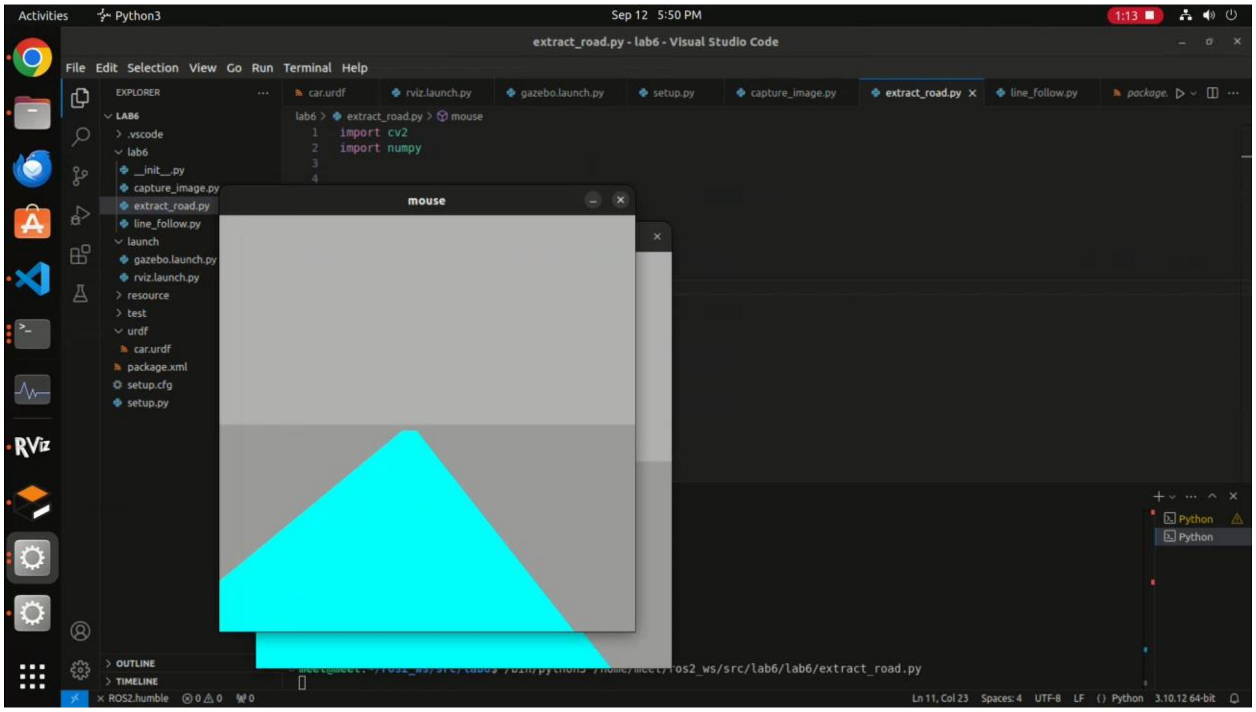
```
meet@meet:~$ ros2 run lab6 capture
```

2. Extracting road from image:

```
import cv2
import numpy
image = cv2.imread('/home/meet/shot.png')
def mouse(event,x,y,flags,param):
    if event==cv2.EVENT_LBUTTONDOWN:
        h=image[y,x,0]
        s=image[y,x,1]
        v=image[y,x,2]
        print("H:",h)
        print("S:",s)
        print("V:",v)
cv2.namedWindow('mouse')
cv2.setMouseCallback('mouse',mouse)
cv2.imshow("original image", image)
cv2.imshow("mouse", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
light_line = numpy.array([250,250,0])
dark_line = numpy.array([255,255,10])
mask = cv2.inRange(image, light_line,dark_line)
cv2.imshow('mask', mask)
cv2.waitKey(0)
cv2.destroyAllWindows()
canny= cv2.Canny(mask,30,5)
cv2.imshow('edge', canny)
cv2.waitKey(0)
cv2.destroyAllWindows()
print(canny.shape)
r1=200;c1=0
img = canny[r1:r1+200,c1:c1+512]
cv2.imshow('crop', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
edge=[]
row =150

for i in range (512):
    if(img[row,i]==255):
        edge.append(i)
print(edge)


if(len(edge)==4):
    left_edge=edge[0]
    right_edge=edge[2]
    print(edge)
if(len(edge)==3):
    if(edge[1]-edge[0] > 5):
        left_edge=edge[0]
        right_edge=edge[1]
    else:
        left_edge=edge[0]
        right_edge=edge[2]
road_width=(right_edge-left_edge)
frame_mid = left_edge + (road_width/2)
mid_point = 512/2
img[row,int(mid_point)]=255
print(mid_point)
error=mid_point-frame_mid
if(error < 0):
    action="Go Right"
else :
    action="Go Left"
print("error", error)
img[row,int(frame_mid)]=255
print("mid point of the frame", frame_mid)
```

```
f_image = cv2.putText(img, action, (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0), 1, cv2.LINE_AA)
cv2.imshow('final image',f_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

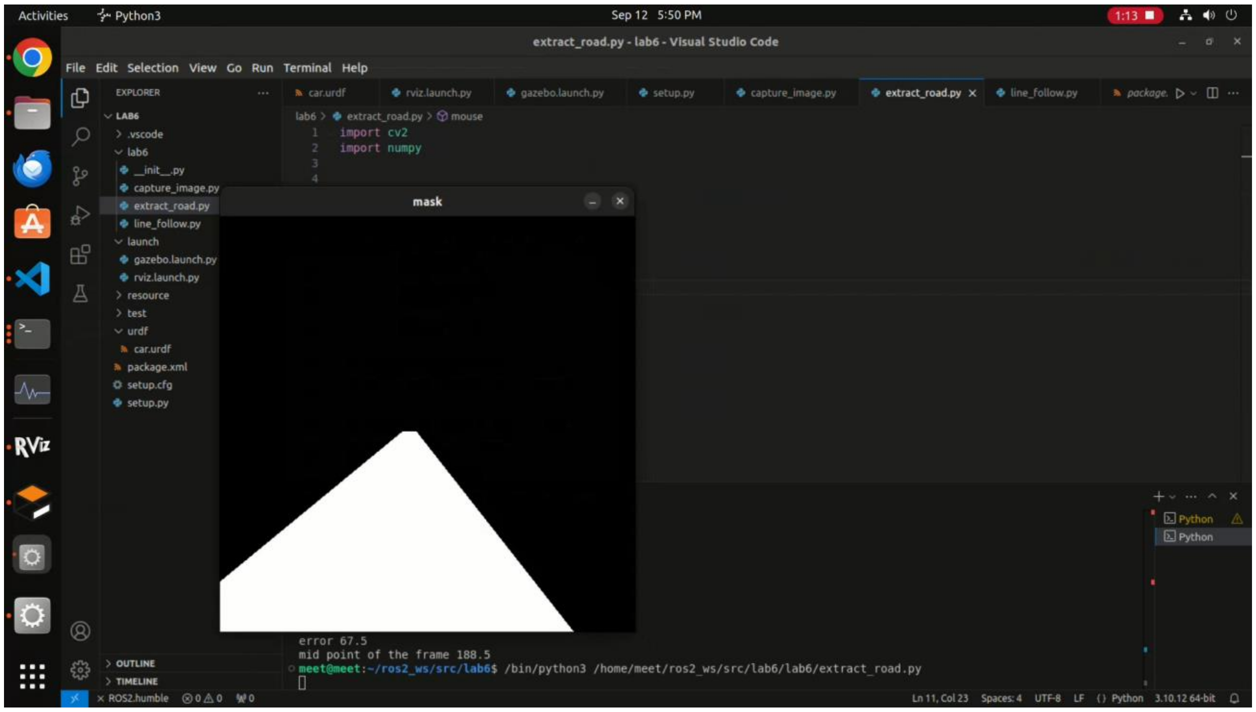
- Image



```
import cv2
import numpy
```

```
meet@meet:~/ros2_ws/src/lab6$ /bin/python3 /home/meet/ros2_ws/src/lab6/lab6/extract_road.py
```

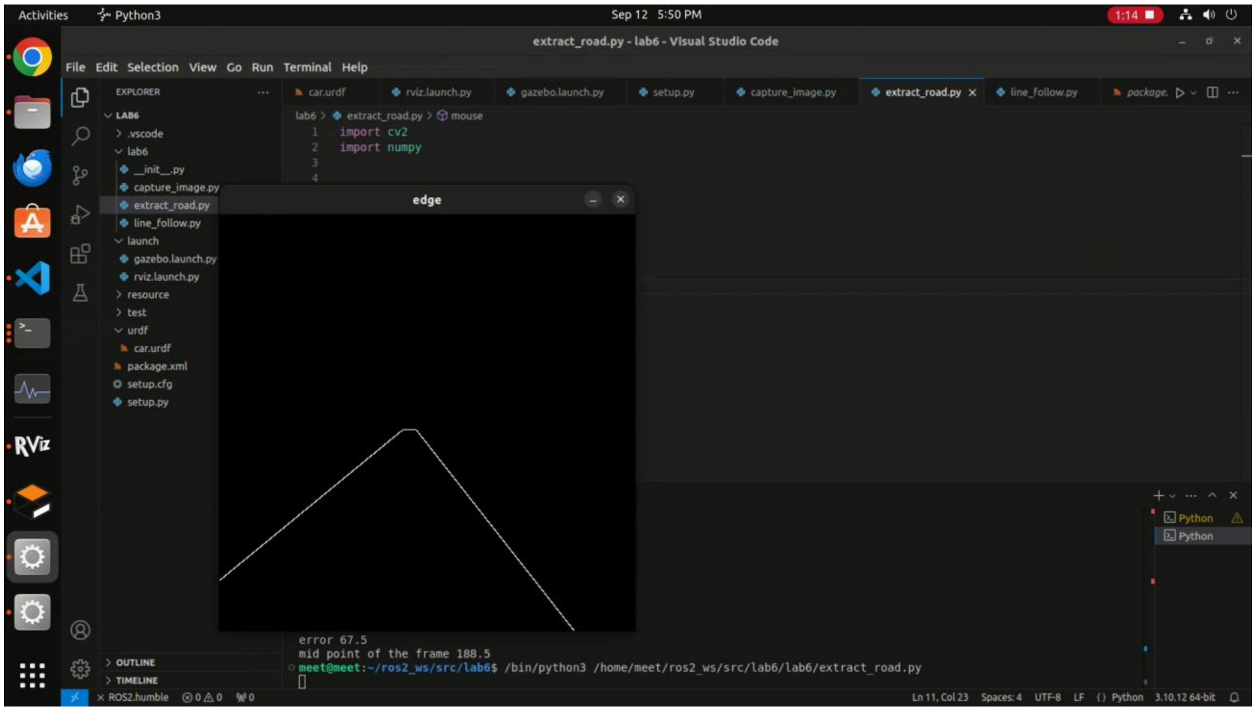
- Mask



```
import cv2
import numpy
```

```
error 67.5
mid point of the frame 188.5
meet@meet:~/ros2_ws/src/lab6$ /bin/python3 /home/meet/ros2_ws/src/lab6/lab6/extract_road.py
```

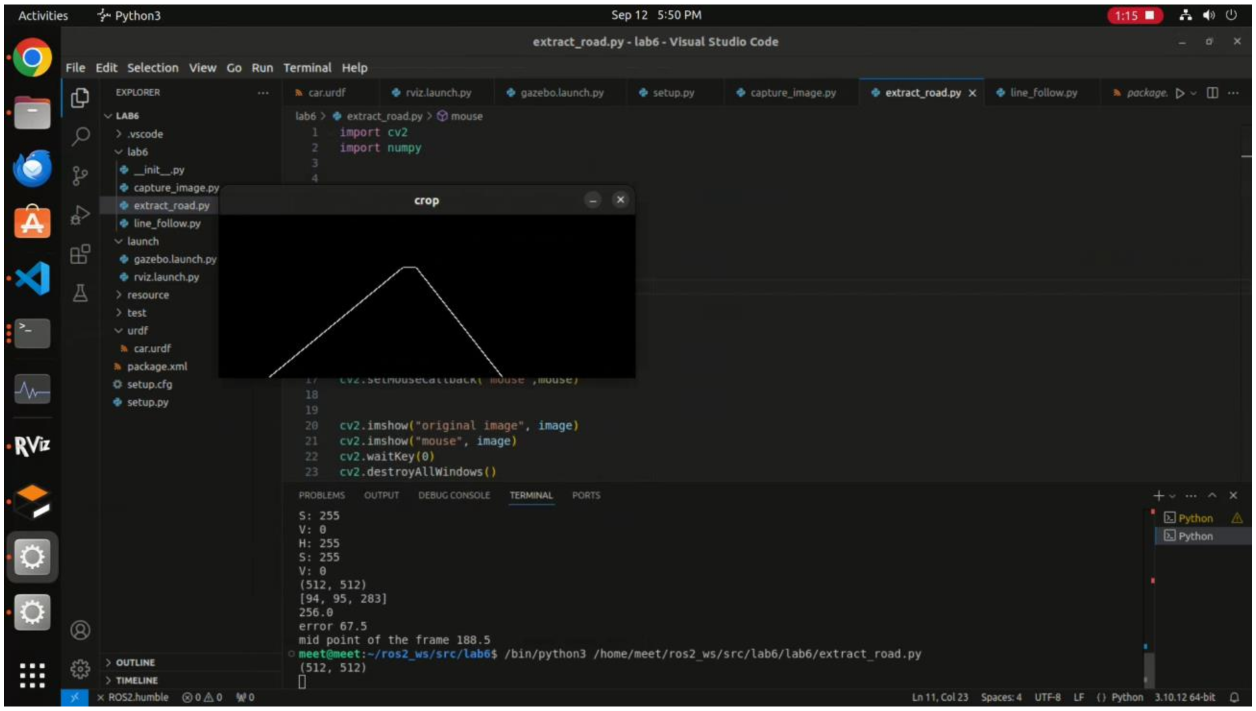
- Edge



```
import cv2
import numpy
```

```
error 67.5
mid point of the frame 188.5
meet@meet:~/ros2_ws/src/lab6$ /bin/python3 /home/meet/ros2_ws/src/lab6/lab6/extract_road.py
```

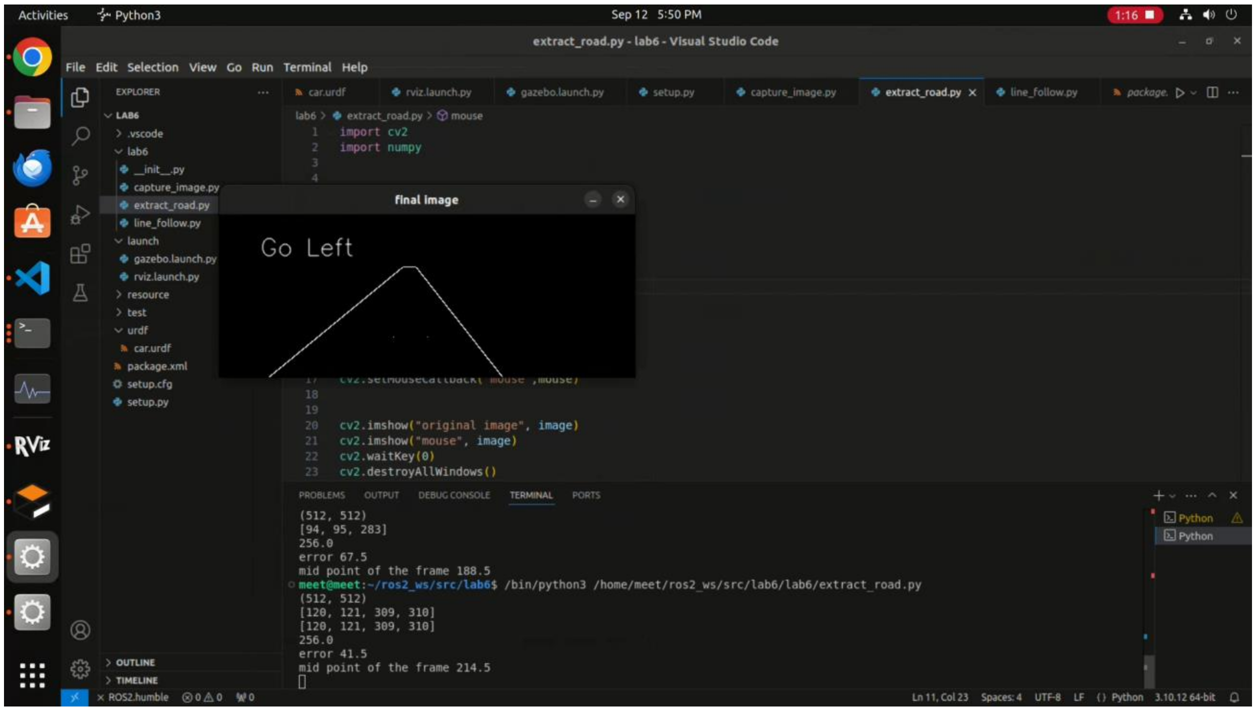
- Crop



```
import cv2
import numpy

cv2.setMouseCallback('mouse',mouse)


cv2.imshow("original image", image)
cv2.imshow("mouse", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
S: 255
V: 0
H: 255
S: 255
V: 0
(512, 512)
[94, 95, 283]
256.0
error 67.5
mid point of the frame 188.5
meet@meet:~/ros2_ws/src/lab6$ /bin/python3 /home/meet/ros2_ws/src/lab6/lab6/extract_road.py
(512, 512)
```

- Directions



```
import cv2
import numpy

cv2.setMouseCallback('mouse',mouse)


cv2.imshow("original image", image)
cv2.imshow("mouse", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

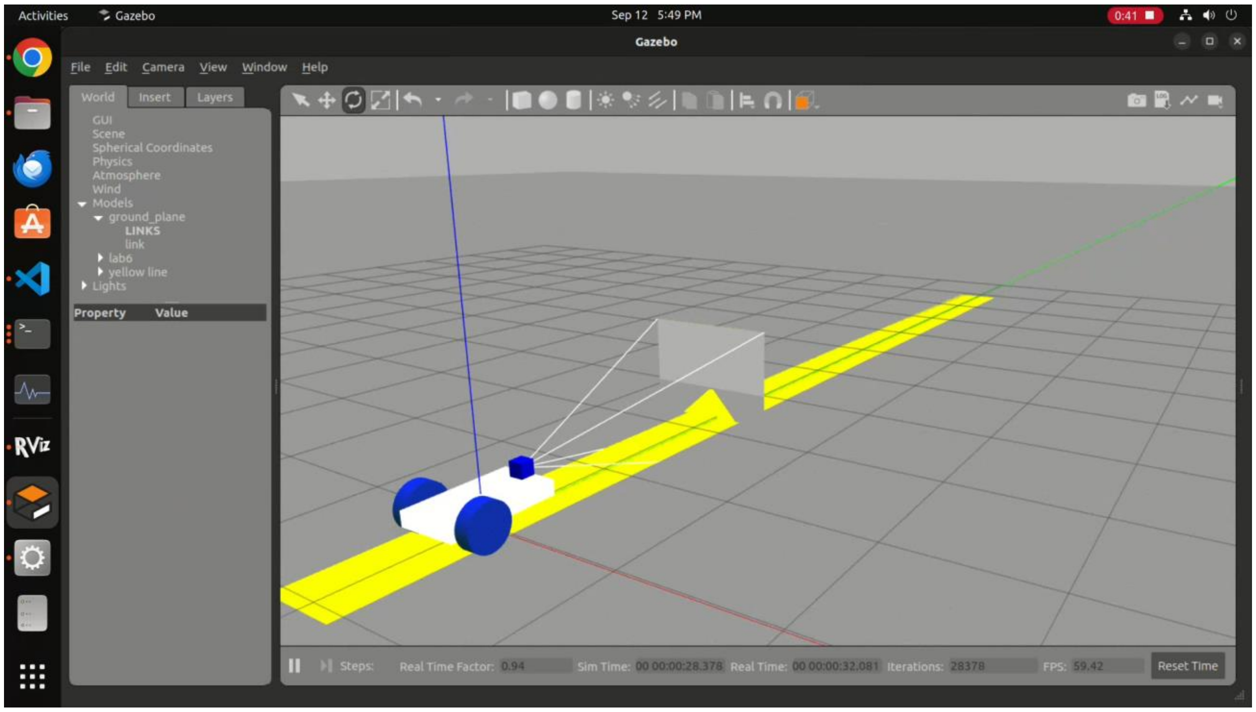
```
(512, 512)
[94, 95, 283]
256.0
error 67.5
mid point of the frame 188.5
meet@meet:~/ros2_ws/src/lab6$ /bin/python3 /home/meet/ros2_ws/src/lab6/lab6/extract_road.py
(512, 512)
[120, 121, 309, 310]
[120, 121, 309, 310]
256.0
error 41.5
mid point of the frame 214.5
```

3. Line following:

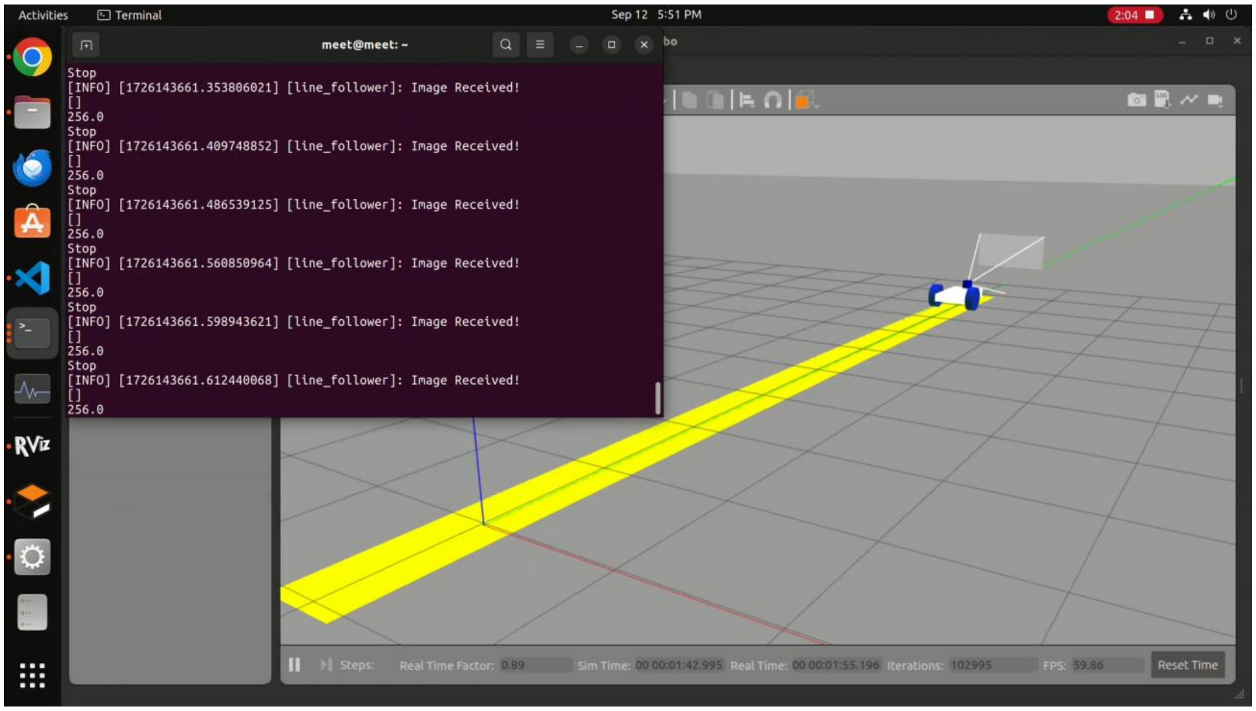
```
#!/usr/bin/env python3

import sys
import cv2
import numpy
import rclpy
from rclpy.node import Node
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist

class LineFollower(Node):
    def __init__(self):
        super().__init__('line_follower')
        self.bridge = CvBridge()
        self.subscriber = self.create_subscription(Image,'/camera1/image_raw',self.process_data,
10)
        self.publisher = self.create_publisher(Twist, '/cmd_vel', 40)
        timer_period = 0.2
        self.timer = self.create_timer(timer_period, self.send_cmd_vel)
        self.velocity=Twist()
        self.empty = False
        self.error = 0
        self.action=""
        self.get_logger().info("Node Started!")

    def send_cmd_vel(self):

        if(self.empty):
            self.velocity.linear.x=0.0
            self.velocity.angular.z= 0.0
            self.action="Stop"

        else:
            if(self.error > 0):
```

```python
                self.velocity.linear.x=0.1
                self.velocity.angular.z=0.3
                self.action="Go Left"
            elif(self.error < 0):
                self.velocity.linear.x=0.1
                self.velocity.angular.z=-0.3
                self.action="Go Right"
            elif(self.error==0):
                self.velocity.linear.x=0.1
                self.velocity.angular.z= 0.0
                self.action="Go Straight"


        self.publisher.publish(self.velocity)


    ## Subscriber Call Back

    def process_data(self, data):
        self.get_logger().info("Image Received!")
        frame = self.bridge.imgmsg_to_cv2(data)
        light_line = numpy.array([250,250,0])
        dark_line = numpy.array([255,255,10])
        mask = cv2.inRange(frame, light_line,dark_line)
        cv2.imshow('mask', mask)


        canny= cv2.Canny(mask,30,5)
        cv2.imshow('edge', canny)


        r1=200;c1=0
        img = canny[r1:r1+200,c1:c1+512]
        cv2.imshow('crop', img)


        edge=[]
        row =150

        for i in range(512):
            if(img[row,i]==255):
                edge.append(i)
        print(edge)

        if(len(edge)==0):
            left_edge=512//2
            right_edge=512//2
            self.empty = True

        if(len(edge)==1):
            if edge[0]>512//2:
                left_edge=0
                right_edge=edge[0]
                self.empty = False
            else:
                left_edge=edge[0]
                right_edge=512
                self.empty = False


        if(len(edge)==2):
            left_edge=edge[0]
            right_edge=edge[1]
            self.empty = False

        if(len(edge)==3):
            if(edge[1]-edge[0]>5):
                left_edge=edge[0]
                right_edge=edge[1]
                self.empty = False
            else:
                left_edge=edge[0]
                right_edge=edge[2]
                self.empty = False
        if(len(edge)==4):
            left_edge=edge[0]
            right_edge=edge[2]
            self.empty = False

        if(len(edge)>=5):
            left_edge=edge[0]
            right_edge=edge[len(edge)-1]
            self.empty = False
        road_width=(right_edge-left_edge)
        frame_mid = left_edge + (road_width/2)
        mid_point = 512/2
        img[row,int(mid_point)]=255
        print(mid_point)
        self.error=mid_point-frame_mid
        img[row,int(frame_mid)]=255
        print(self.action)
        f_image = cv2.putText(img, self.action, (100,100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,),
2, cv2.LINE_AA)

def main(args=None):
  rclpy.init(args=args)
  node = LineFollower()
  rclpy.spin(node)
  rclpy.shutdown()

if __name__ == '__main__':
  main()
```

- Start position



- End position



```
Stop
[INFO] [1726143661.353806021] [line_follower]: Image Received!
[]
256.0
Stop
[INFO] [1726143661.409748852] [line_follower]: Image Received!
[]
256.0
Stop
[INFO] [1726143661.486539125] [line_follower]: Image Received!
[]
256.0
Stop
[INFO] [1726143661.560850964] [line_follower]: Image Received!
[]
256.0
Stop
[INFO] [1726143661.598943621] [line_follower]: Image Received!
[]
256.0
Stop
[INFO] [1726143661.612440068] [line_follower]: Image Received!
[]
256.0
```

Conclusion The robot successfully followed the yellow line using OpenCV for real-time image processing in Gazebo.