

## Lab6: ROS2 Perception – Line Follower

- How to use OpenCV in ROS2
- How to follow a line
- How to find different elements based on color
- Track multiple paths and decide
- Create a basic PID for the line following

OpenCV is the most extensive and complete library for image recognition. With it, you can work with images like never before: applying filters, post-processing, and working with images in any way you want. OpenCV is not a ROS2 library, but it's been integrated nicely into ROS with [http://wiki.ros.org/cv\\_bridge](http://wiki.ros.org/cv_bridge). This package allows the ROS imaging topics to use the OpenCV image variable format.

For example, OpenCV images come in BGR image format, while regular ROS images are in the more standard RGB encoding. OpenCV\_bridge provides a nice feature to convert between them. Also, there are many other functions to transfer images to OpenCV variables transparently.

If gazebo is stuck use the following command

```
killall -9 gzserver
```

```
sudo apt install libopencv-dev python3-opencv
```

```
cd ros2_ws/src
```

```
ros2 pkg create lab6 --build-type ament_python --dependencies rclpy std_msgs
```

```
cd ..
```

```
colcon build
```

```
open lab6 with visual studio application
```

```
create urdf folder
```

```
create a file car.urdf inside the urdf folder
```

```
<?xml version="1.0" ?>
```

```
<robot name = "car">
```

```
  <link name="base">
```

```
    <visual>
```

```
    <geometry>
```

```

    <box size="0.75 0.4 0.1"/>
  </geometry>
  <material name="pink">
    <color rgba="1 0 1 1" />
  </material>
</visual>

<inertial>
  <mass value="1" />
  <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01" iyz="0" izz="0.01" />
</inertial>

<collision>
  <geometry>
    <box size="0.75 0.4 0.1"/>
  </geometry>
</collision>

</link>

<link name="wheel_right_link">
  <inertial>
    <mass value="2" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <material name="blue">
      <color rgba="0 0 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
  </collision>
</link>

<joint name="wheel_right_joint" type="continuous">
  <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>

```

```
<parent link="base"/>
<child link="wheel_right_link"/>
<axis xyz="0.0 0.0 1.0"/>
</joint>
```

```
<link name="wheel_left_link">
  <inertial>
    <mass value="2" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <material name="blue">
      <color rgba="0 0 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
  </collision>
</link>
```

```
<joint name="wheel_left_joint" type="continuous">
  <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
  <parent link="base"/>
  <child link="wheel_left_link"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>
```

```
<link name="caster">
  <inertial>
    <mass value="1" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>
```

```

<visual>
  <geometry>
    <sphere radius=".08" />
  </geometry>
  <material name="white" />
</visual>

<collision>
  <origin/>
  <geometry>
    <sphere radius=".08" />
  </geometry>
</collision>
</link>

<joint name="caster_joint" type="continuous">
  <origin xyz="-0.3 0.0 -0.07" rpy="0.0 0.0 0.0"/>
  <axis xyz="0 0 1" />
  <parent link="base"/>
  <child link="caster"/>
</joint>

<link name="camera">
  <inertial>
    <mass value="0.5" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <box size="0.1 0.1 0.1"/>
    </geometry>
    <material name="red">
      <color rgba="1 0 0 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <box size="0.1 0.1 0.1"/>
    </geometry>
  </collision>
</link>

```

```

<joint name="camera_joint" type="fixed">
  <origin xyz="-0.32 0 0.1" rpy="0 0.0 3.14"/>
  <parent link="base"/>
  <child link="camera"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>

```

```

<!--http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials-->
<gazebo reference="base">
  <material>Gazebo/WhiteGlow</material>
</gazebo>
<gazebo reference="wheel_left_link">
  <material>Gazebo/SkyBlue</material>
</gazebo>
<gazebo reference="wheel_right_link">
  <material>Gazebo/SkyBlue </material>
</gazebo>
<gazebo reference="caster">
  <material>Gazebo/Grey</material>
</gazebo>
<gazebo reference="camera">
  <material>Gazebo/Blue</material>
</gazebo>

```

```

<!-- differential robot-->
<gazebo>
  <plugin filename="libgazebo_ros_diff_drive.so"
name="gazebo_base_controller">
    <odometry_frame>odom</odometry_frame>
    <commandTopic>cmd_vel</commandTopic>
    <publish_odom>true</publish_odom>
    <publish_odom_tf>true</publish_odom_tf>
    <update_rate>15.0</update_rate>

    <left_joint>wheel_left_joint</left_joint>
    <right_joint>wheel_right_joint</right_joint>

    <wheel_separation>0.5</wheel_separation>
    <wheel_diameter>0.3</wheel_diameter>

```

```

        <max_wheel_acceleration>0.7</max_wheel_acceleration>
        <max_wheel_torque>8</max_wheel_torque>
        <robotBaseFrame>base</robotBaseFrame>
    </plugin>
</gazebo>

<!-- camera plugin-->

<gazebo reference="camera">
    <sensor type="camera" name="camera1">
        <visualize>true</visualize>
        <update_rate>30.0</update_rate>
        <camera name="head">
            <horizontal_fov>1.3962634</horizontal_fov>
            <image>
                <width>512</width>
                <height>512</height>
                <format>R8G8B8</format>
            </image>
            <clip>
                <near>0.02</near>
                <far>500</far>
            </clip>
        </camera>
        <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
            <alwaysOn>true</alwaysOn>
            <updateRate>0.0</updateRate>
            <cameraName>/camera</cameraName>
            <imageTopicName>image_raw</imageTopicName>
            <cameraInfoTopicName>camera_info</cameraInfoTopicName>
            <frameName>camera_link</frameName>
            <hackBaseline>0.07</hackBaseline>
        </plugin>
    </sensor>
    <material>Gazebo/Blue</material>
</gazebo>

</robot>

```

Create a launch folder

Create a launch file rviz.launch.py

```

import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription

```

```

from launch_ros.actions import Node

def generate_launch_description():
    package_dir = '/home/asha/ros2_ws/src/lab6/urdf'
    urdf = os.path.join(package_dir, 'car.urdf')

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),

        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
            name='joint_state_publisher_gui',
            arguments=[urdf]),

        Node(
            package='rviz2',
            executable='rviz2',
            name='rviz2',
            output='screen'),
    ])

```

Create a launch folder

Create a launch file gazebo.launch.py inside the launch folder

```

import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, ExecuteProcess
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():
    urdf = '/home/asha/ros2_ws/src/lab6/urdf/car.urdf'
    return LaunchDescription([
        # publishes TF for links of the robot without joints
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',

```

```

        output='screen',
        arguments=[urdf]),
# publish TF for Joints only links
Node(
    package='joint_state_publisher',
    executable='joint_state_publisher',
    name='joint_state_publisher',
    output='screen',
),
# open gazebo
ExecuteProcess(
    cmd=['gazebo', '--verbose', '-s', 'libgazebo_ros_factory.so'],
    output='screen'),
Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    name='urdf_spawner',
    output='screen',
    arguments=["-topic", "/robot_description", "-entity", "lab6"])
])

```

- **Height and width:** These are the dimensions in camera pixels. In this case, it's 512 x 512.
- **Encoding:** How these pixels are encoded. This means what each value in the data array will mean. In this case, it's **rgb8**. This means that the data values will be a color value represented as red/green/blue in 8-bit integers.
- **Data:** The image data.

#### Terminal 1:

```

ros2 launch lab6 rviz.launch.py
ros2 launch lab6 gazebo.launch.py

```

ctrl + c to kill the rviz and gazebo window

create a file capture\_image.py inside the lab 6 folder

```

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/asha/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/asha/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()

```

Terminal 1:

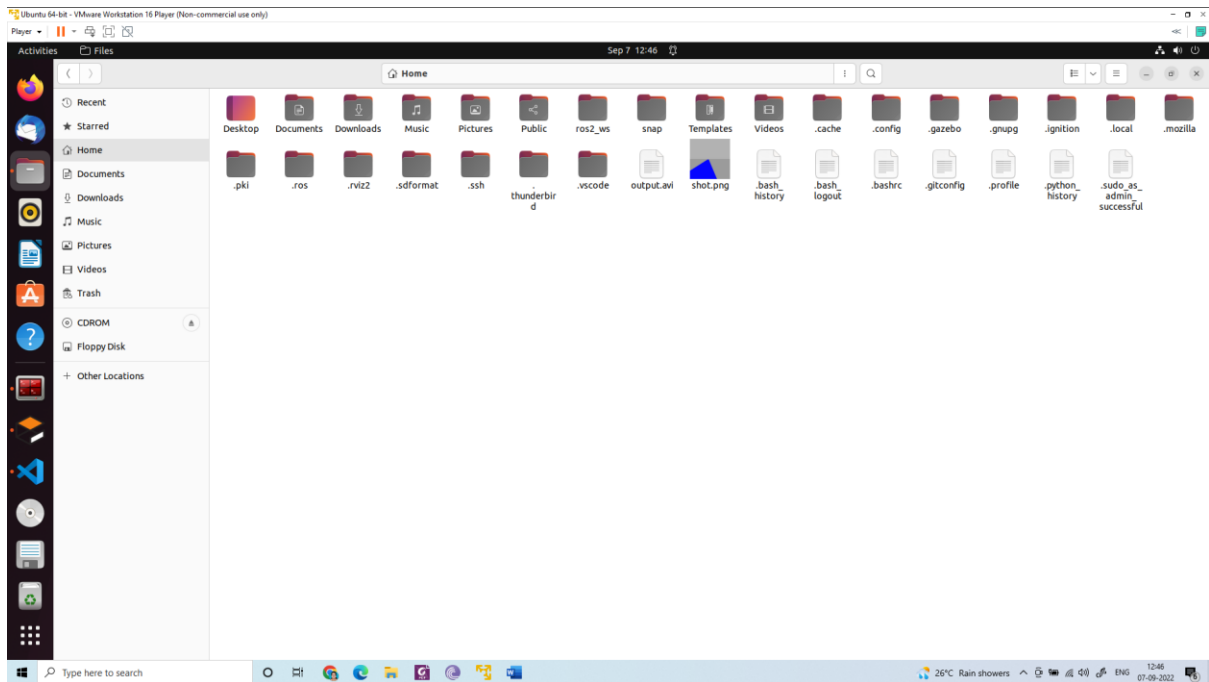
`ros2 launch lab6 gazebo.launch.py`

draw a line and place the robot on the line (insert → yellow line)

Terminal 2:

`ros2 run lab6 capture`

Press ctrl + c to capture the image of the road



create a file `extract_road.py` inside the lab 6 folder

```
import cv2
import numpy
```

```
image = cv2.imread('/home/asha/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([250,0,0])
dark_line = numpy.array([255,10,10])
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

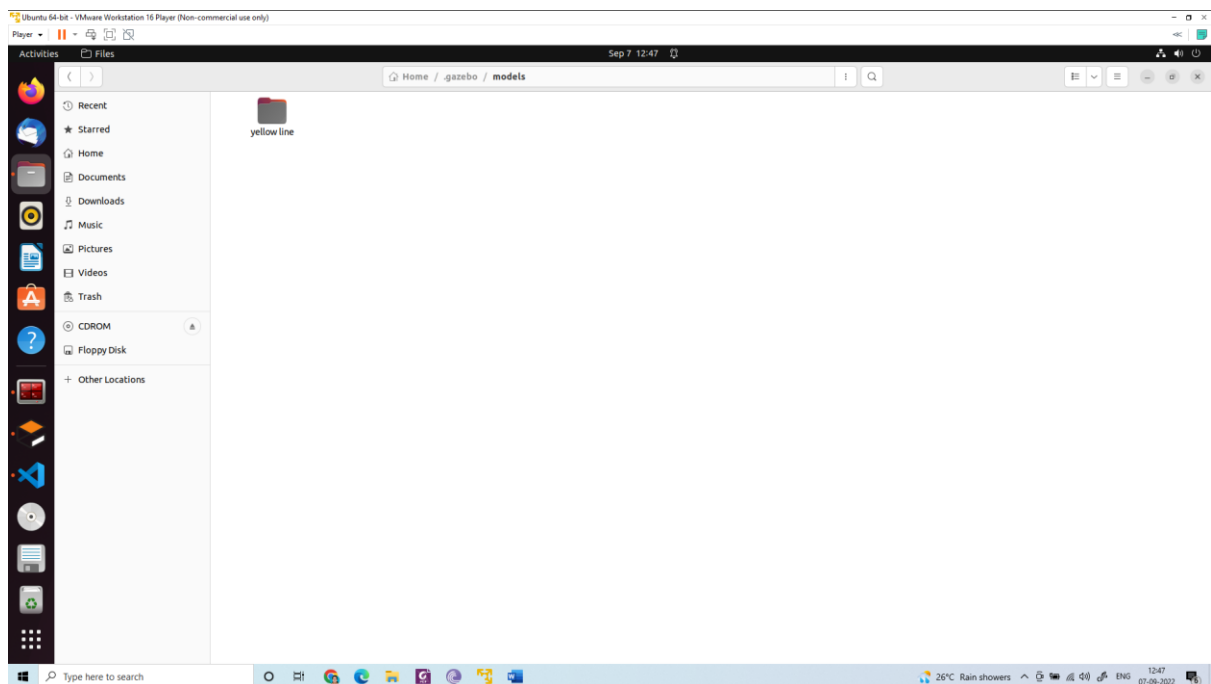
if(error < 0):
    action="Go Right"
else :
    action="Go Left"

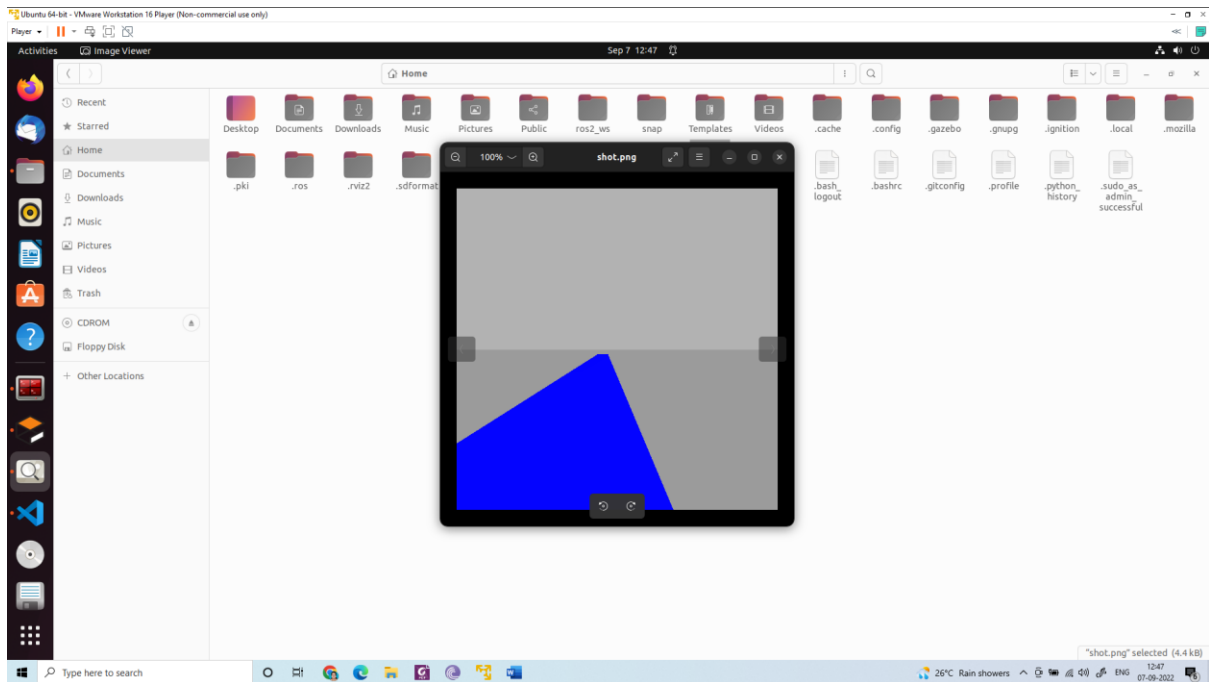
print("error", error)

```

```
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()
```





Create a line\_follow.py inside the folder lab 6

```
#!/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):
        self.velocity.linear.x=0.1
        self.velocity.angular.z=0.1
        self.action="Go Left"
    elif(self.error < 0):
        self.velocity.linear.x=0.1
        self.velocity.angular.z=-0.1
        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([250,0,0])
    dark_line = numpy.array([255,10,10])
    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()

```

edit setup.py file

```

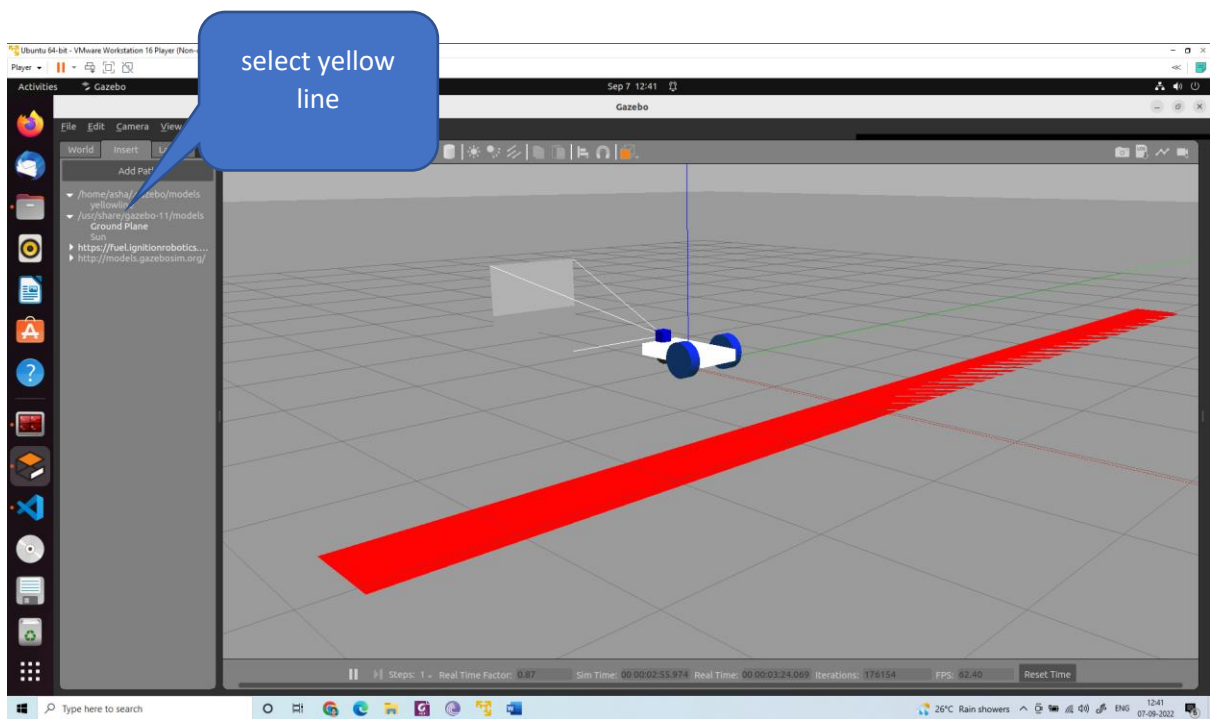
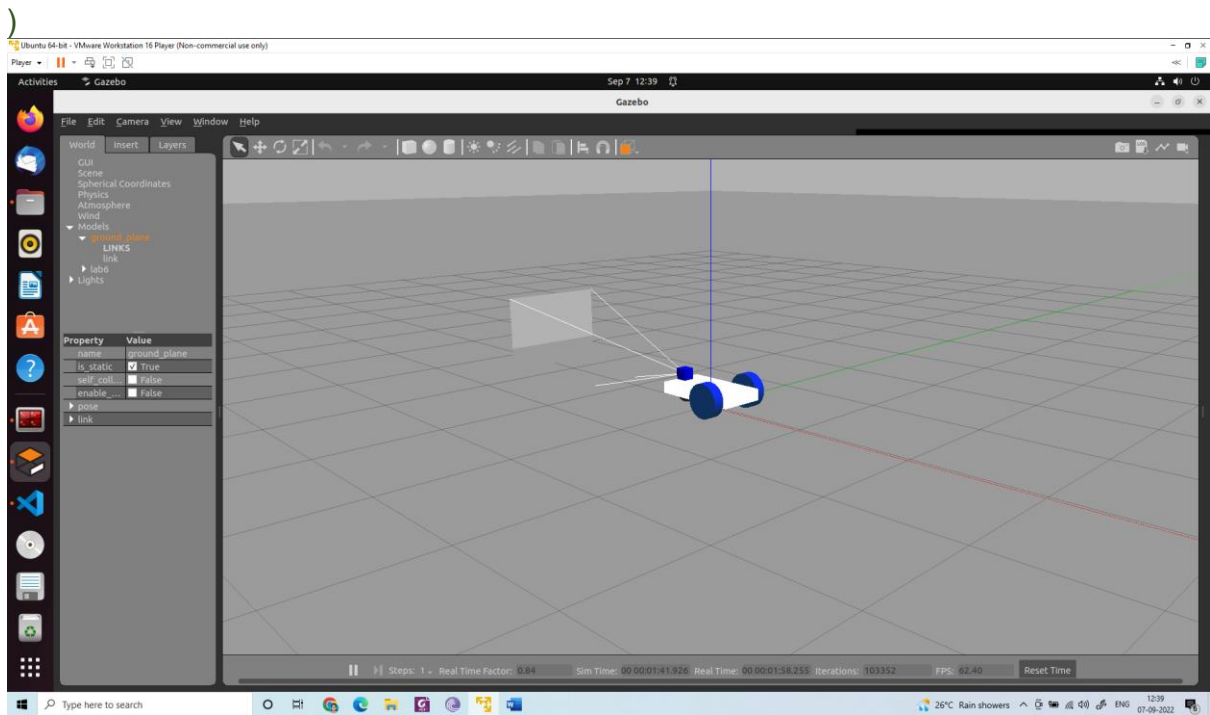
from setuptools import setup
import os
from glob import glob

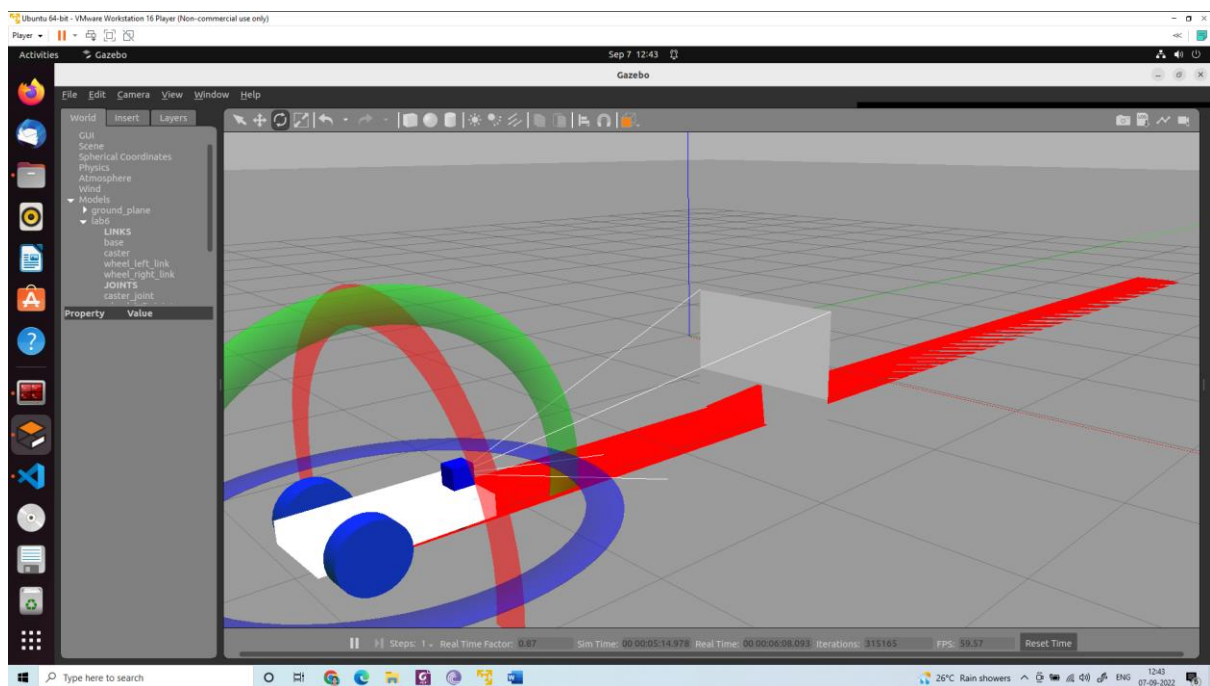
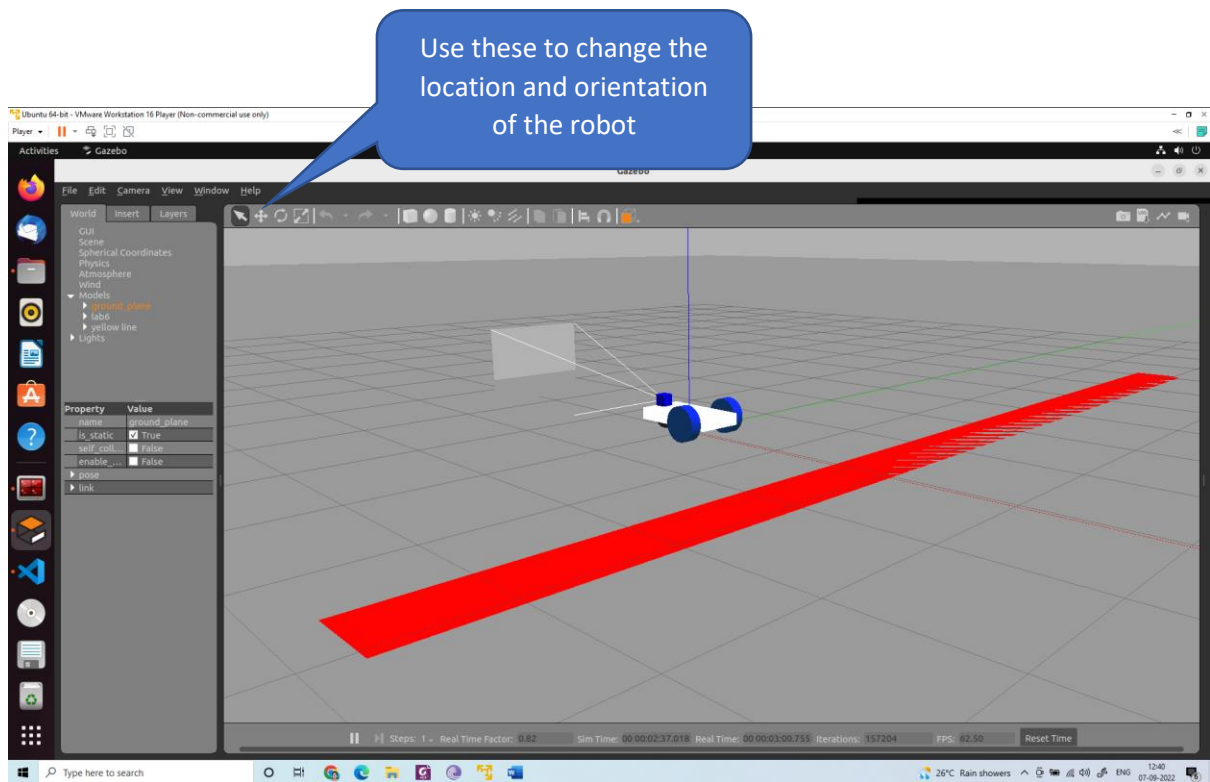
package_name = 'lab6'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launch/*')),
        (os.path.join('share', package_name), glob('urdf/*'))
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha.cs12@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'capture = lab6.capture_image:main',
            'line = lab6.line_follow:main'
        ],
    },
)

```



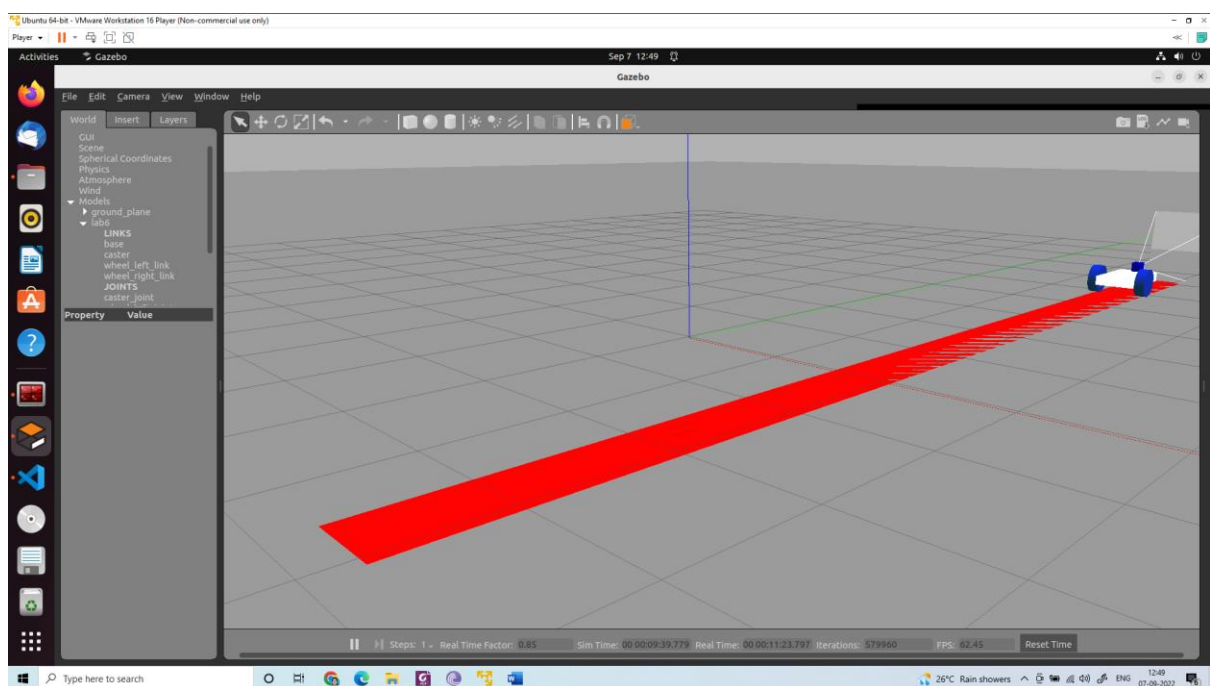
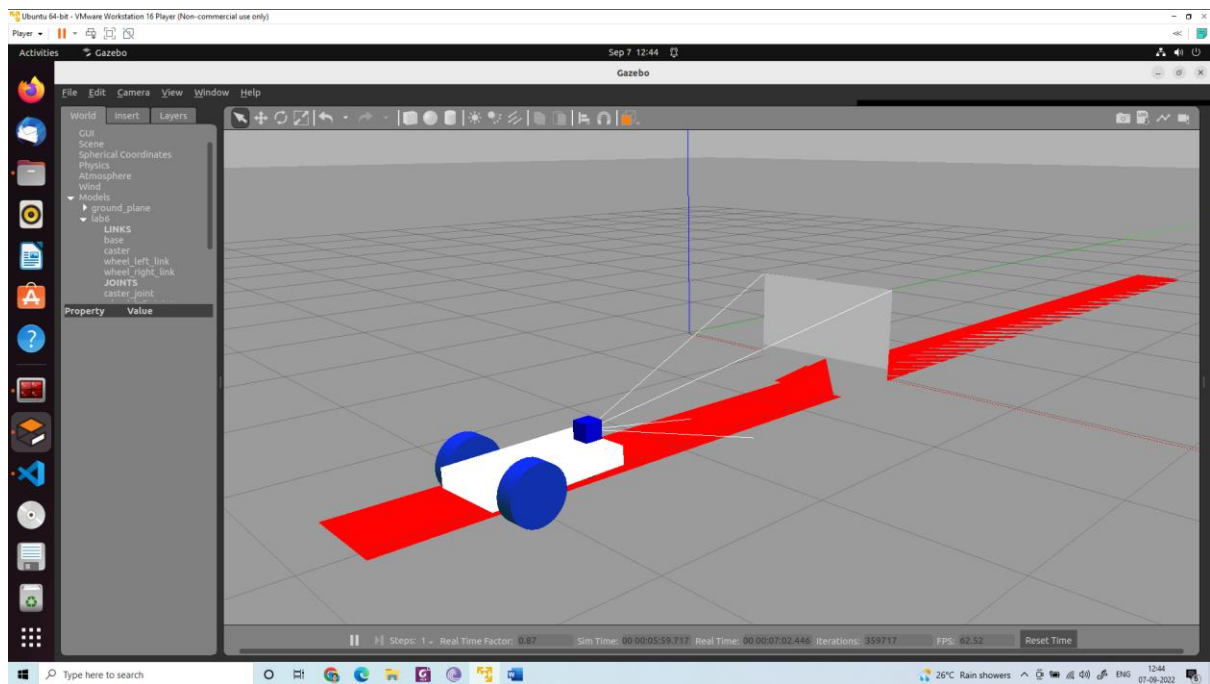




Terminal 1:  
`ros2 launch lab6 gazebo.launch.py`

Terminal 2:  
`ros2 topic list`

Terminal 3:  
`ros2 run lab6 line`



Create a red line

Create a world folder

Create a yellow line folder

## Supplementary material:

### Create a model.config

```
<?xml version="1.0" ?>
<model>
  <name>yellowline</name>
  <version>1.0</version>
  <sdf version="1.6">model.sdf</sdf>
  <author>
    <name></name>
    <email></email>
  </author>
  <description></description>
</model>
```

### Create a model.sdf

```
<?xml version="1.0"?>
<sdf version="1.6">
<model name="yellow line">
  <static>true</static>
  <link name="link_ground">
    <collision name="collision">
      <geometry>
        <plane>
          <normal>0 0 1</normal>
          <size>0.1 3.2</size>
        </plane>
      </geometry>

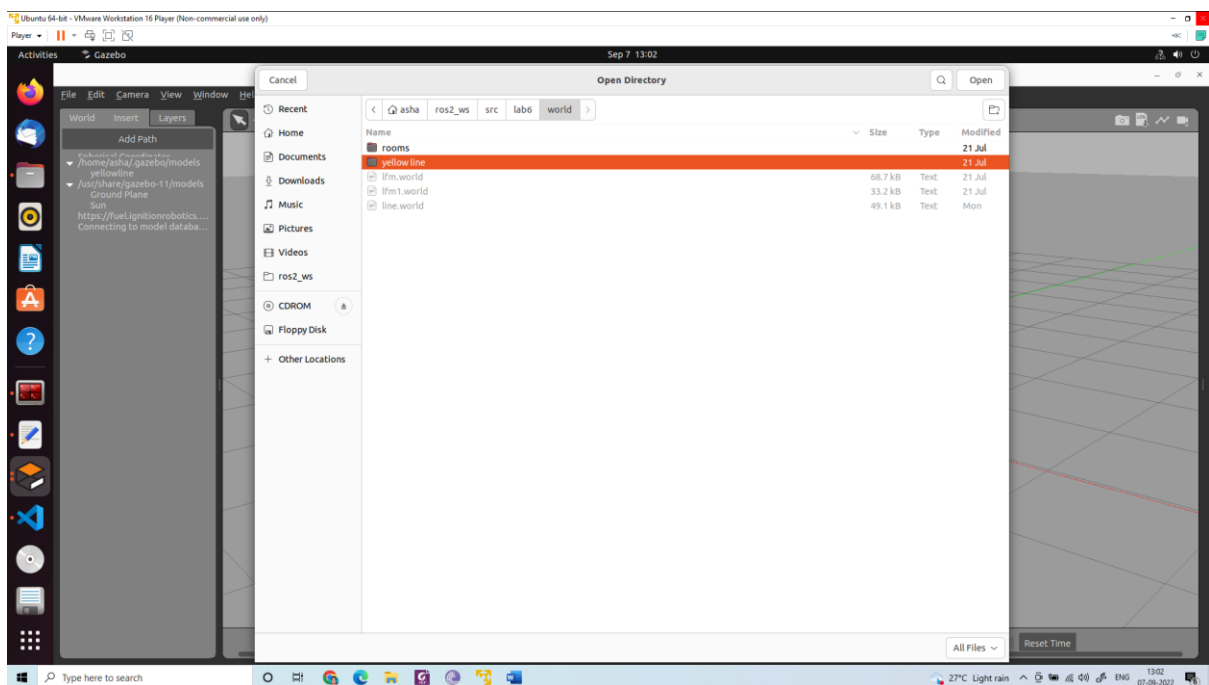
      <surface>
        <friction>
          <ode>
            <mu>100</mu>
            <mu2>50</mu2>
          </ode>
        </friction>
      </surface>
    </collision>
    <visual name="visual_ground">
      <cast_shadows>false</cast_shadows>
      <geometry>
        <plane>
          <normal>0 0 1</normal>
          <size>0.5 10</size>
        </plane>
      </geometry>
      <material>
        <script>
```

```

    <uri>file://media/materials/scripts/gazebo.material</uri>
    <name>Gazebo/Red</name>
  </script>
</material>
</visual>
</link>
</model>
</sdf>

```

Copy the folder into /home/asha/.gazebo/models/  
 In the gazebo window (gazebo)  
 Insert→/home/asha/.gazebo/models/→yellow line→all files  
 close gazebo (killall 9 gzserver) and open again (gazebo)



Insert→/home/asha/.gazebo/models/→yellow line

Exercise: Write a code to track the red ball in the gazebo simulation.