Edge Detection

1. Content Description

This course explains how to use depth imaging for edge detection, combined with the chassis, to enable the robot to stop at edges and prevent falls. This can be expanded to include depth imaging obstacle avoidance functionality. This section requires entering commands in a terminal. The terminal you open depends on your board type. This lesson uses the Raspberry Pi as an example.

For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to **[01. Robot Configuration and Operation Guide] -- [4.Enter Docker (For JETSON Nano and RPi 5)]**.

For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

2. Program Startup

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

Enter the Docker container (see [Docker course] --- [4. Docker Startup Script] for steps).

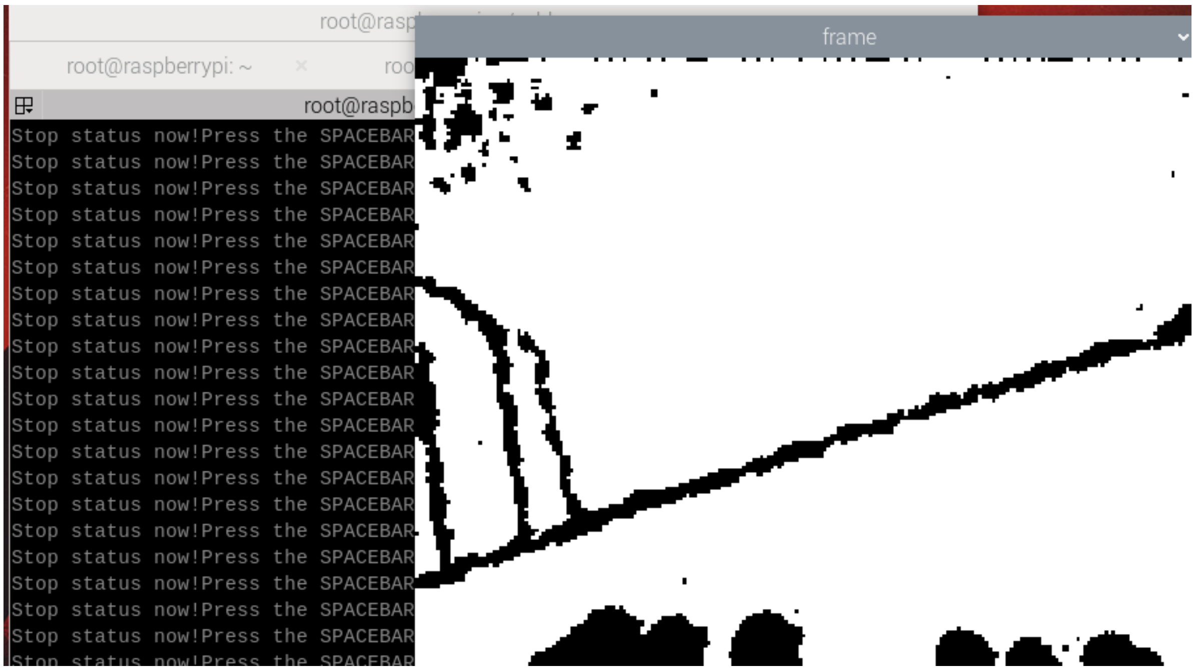
All the following commands must be executed within the same Docker container **(see [Docker course] --- [3. Docker Submission and Multi-Terminal Access] for steps).**

First, enter the following command in the terminal to start the camera.

```
ros2 launch ascamera hp60c.launch.py
```

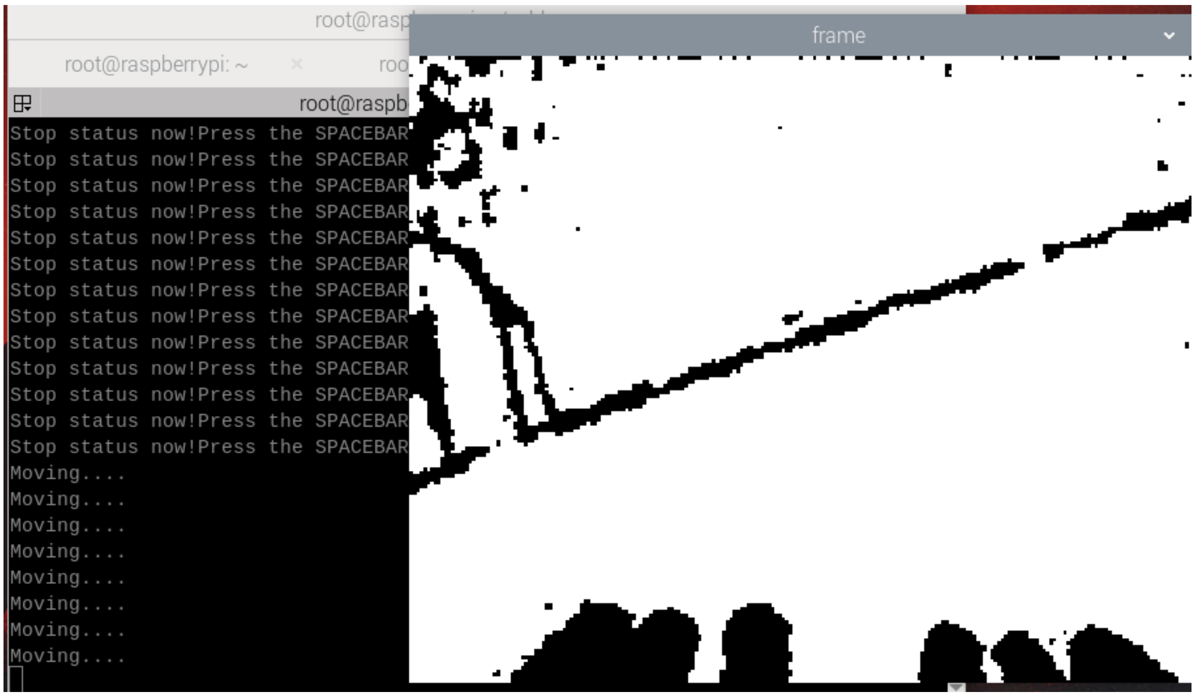
After the camera successfully starts, open another terminal and enter the following command to start the edge detection program.

```
ros2 run yahboomcar_depth Edge_Detection
```

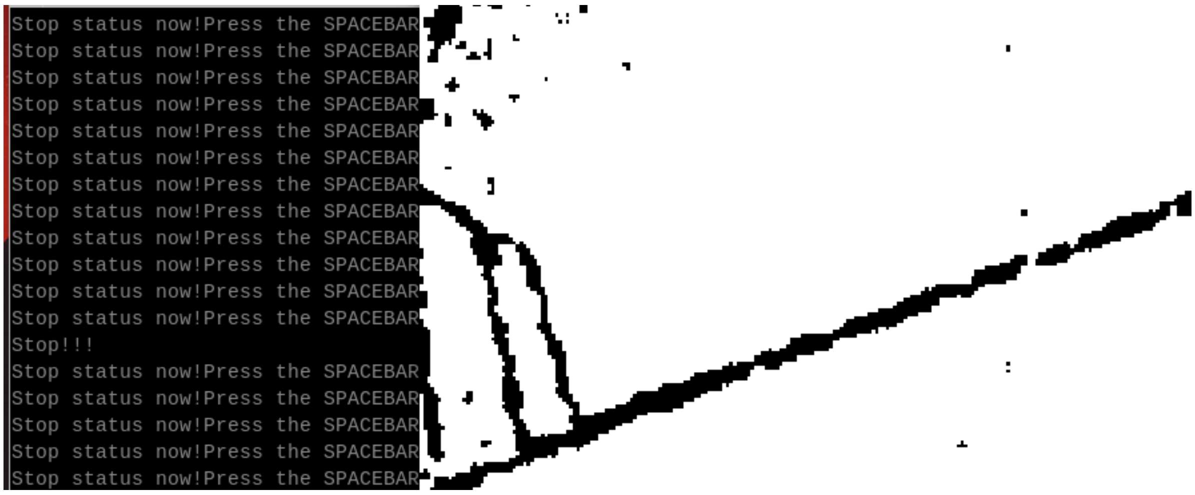


```
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
```

As shown in the image above, after the program starts, it will print the current status, indicating that it is stopped. Pressing the spacebar changes the status. After pressing the spacebar, if the robot does not detect an edge, it will move forward and print "Moving..."; if it detects an edge, it will stop and print "Stop!!!".



```
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Moving....
Moving....
Moving....
Moving....
Moving....
Moving....
Moving....
Moving....
```



```
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop!!!
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
Stop status now!Press the SPACEBAR
```

3. Core Code

Program Code Path:

- Raspberry Pi 5 and Jetson Nano Board

  The program code is running in Docker. The path in Docker is /root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Advanced/Edge_Detection.py
- Orin board

  The program code path is /home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Advanced/Edge_Detection.py
- RDK X5

  The program code path is /home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Advanced/Edge_Detection.py

Import the necessary library files.

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from geometry_msgs.msg import Twist
from arm_msgs.msg import ArmJoints
from cv_bridge import CvBridge
import cv2
import numpy as np
import threading
```

Depth image decoding format,

```
encoding = ['16UC1', '32FC1']
```

Initialize variables and define publishers and subscribers,

```
def __init__(self, name):
    super().__init__(name)
    self.pub_vel = self.create_publisher(Twist,'/cmd_vel',1)
    self.sub_depth =
self.create_subscription(Image,"/ascamera_hp60c/camera_publisher/depth0/image_raw",self.get_DepthImageCallBack,100)
    #The forward speed of the car
    self.lin_x = 0.1
    self.Joy_active = False
    self.depth_bridge = CvBridge()
    self.move_flag = False
```

Depth image topic callback function, and calculate the center point depth distance information,

```
def get_DepthImageCallBack(self,msg):
    depth_image = self.depth_bridge.imgmsg_to_cv2(msg, encoding[1])
    #The calling thread passes in the acquired depth image and calculates the
depth information
```

```
        compute_ = threading.Thread(target=self.compute_dist, args=(depth_image,))
        compute_.start()
        compute_.join()
        key = cv2.waitKey(10)
        if key == 32:
            self.move_flag = True
        cv2.imshow("frame", depth_image)

    def compute_dist(self,result_frame):
        frame = cv2.resize(result_frame, (640, 480))
        depth_image_info = frame.astype(np.float32)
        #Determine whether the current state is moving. If so, determine the distance to the center point. If not, call the function, issue a parking instruction, and print information.
        if  self.move_flag == True:
            #Determine whether the depth information of the center point, that is, the point x=320, y=240, is greater than 0.3m. If so, call the function to issue a stop command. If not, call the function to issue a forward command.
            if depth_image_info[240, 320]/1000>0.3:
                self.pubVel(0,0,0)
                self.move_flag = False
                print("Stop!!!")
            else:
                self.pubVel(self.lin_x,0,0)
                print("Moving....")
        else:
            self.pubVel(0,0,0)
            print("Stop status now!Press the SPACEBAR to change the state.")
```

Release speed function,

```
    def pubVel(self,vx,vy,vz):
        vel = Twist()
        vel.linear.x = float(vx)
        vel.linear.y = float(vy)
        vel.angular.z = float(vz)
        self.pub_vel.publish(vel)
```

Three variables are passed in: the speed in the x direction, the speed in the y direction, and the angular velocity. After assigning the values, call `self.pub_vel.publish(vel)` to publish the speed topic.