Depth camera distance measurement

1. Content Description

This lesson uses a depth camera to measure distance within its range. This lesson requires entering commands in a terminal.

This section requires entering commands in the terminal. The terminal you open depends on your motherboard type. This lesson uses the Raspberry Pi 5 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 section in the terminal. For instructions on entering the Docker container from the host computer, refer to this product tutorial **[Configuration and Operation Guide]--[Enter the Docker (Jetson Nano and Raspberry Pi 5 users, see here)]**.

Simply open the terminal on the Orin motherboard and enter the commands mentioned in this section.

2. Program startup

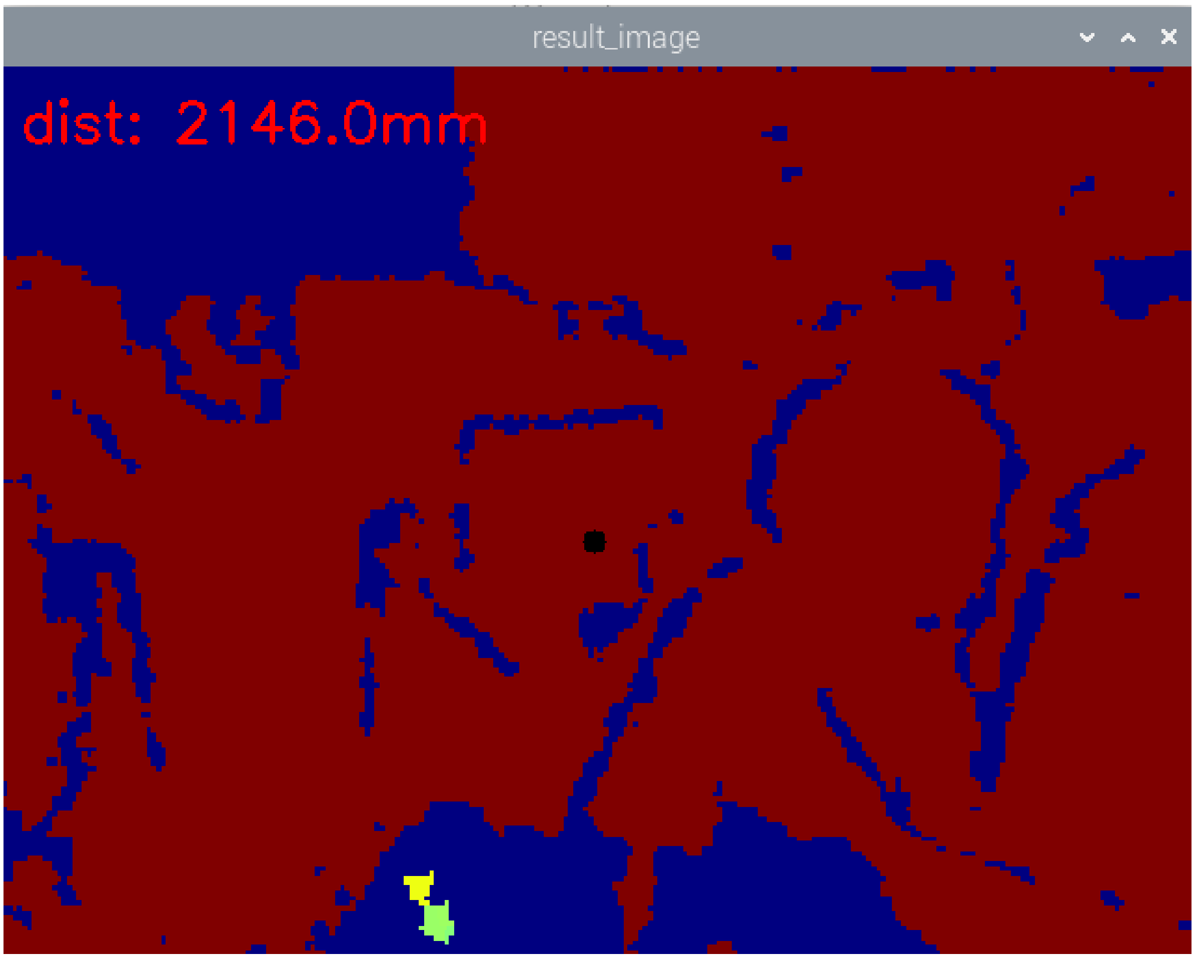
First, in the terminal, enter the following command to start the camera,

```
ros2 launch orbbec_camera dabai_dcw2.launch.py
```

After successfully starting the camera, open another terminal and enter the following command in the terminal to start the deep pseudo-color image conversion program:

```
ros2 run yahboom_M3Pro_DepthCam Measure_Distance
```

After startup, as shown below,



Click the red area with the mouse to select the point to be measured. The selected point will turn black and the distance measured by the depth camera will be printed directly in the upper left corner of the image in millimeters.

3. Core code analysis

Program code path:

- Raspberry Pi 5 and Jetson-Nano board

  The program code is in the running docker. The path in docker is `/root/yahboomcar_ws/src/yahboom_M3Pro_DepthCam/yahboom_M3Pro_DepthCam/Measure_Distance.py`

- Orin Motherboard

  The program code path is `/home/jetson/yahboomcar_ws/yahboom_M3Pro_DepthCam/yahboom_M3Pro_DepthCam/Measure_Distance.py`

Import the library files used

```
from cv_bridge import CvBridge
import cv2
from rclpy.node import Node
import rclpy
from sensor_msgs.msg import Image
import numpy as np
```

Defines the depth image decoding format

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

Define subscribers and define deep image topics

```
self.sub_depth =
self.create_subscription(Image,"/camera/depth/image_raw",self.get_DepthImgCallBa
ck,100)
```

Define self.depth_bridge to convert the message format into an image format that openc can handle

```
self.depth_bridge = CvBridge()
```

Convert image data message into image

```
depth_image = self.depth_bridge.imgmsg_to_cv2(depth_frame, encoding[1])
```

Call the opencv image processing function cv2.applyColorMap to convert the depth map

```
depth_to_color_image = cv2.applyColorMap(cv2.convertScaleAbs(depth_image,
alpha=1.0), cv2.COLORMAP_JET)
```

Convert the image data Numpy array into a 32-bit single-precision floating-point array. This step ensures that floating-point precision is used in subsequent calculations.

```
depth_image_info = depth_image.astype(np.float32)
```

Get the depth information of a point. In a 2D image, use (x, y) to determine the position of a point.

```
dist = depth_image_info[self.y,self.x]
```

Process the depth information and draw it into the image.

```
dist = round(dist,3)
dist = 'dist: ' + str(dist) + 'mm'
cv2.putText(depth_to_color_image, dist,  (10, 40), cv2.FONT_HERSHEY_SIMPLEX,
1.0, (0, 0, 255), 2)
```

Call the opecv function to get the mouse click event, click a point in the valid area of the image, and get the xy value of the point.

```
cv2.setMouseCallback(self.window_name, self.click_callback)
def click_callback(self, event, x, y, flags, params):
    if event == 1:
        self.x = x
        self.y = y
```

Draw the points selected by the mouse on the image and display the image.

```
cv2.circle(depth_to_color_image,(self.x,self.y),1,(0,0,0),10)
cv2.imshow("result_image", depth_to_color_image)
```