Depth Camera Ranging

Before starting this function, you need to close the large program and APP processes. If you need to restart the large program and APP later, start them from the terminal:

```
bash ~/dofbot_pro/APP_DOFBOT_PRO/start_app.sh
```

1. Function Description

After the program starts, by clicking on points in the pseudo-color image with the mouse, within the effective data range of the depth camera, the screen will display the distance between that point and the depth camera.

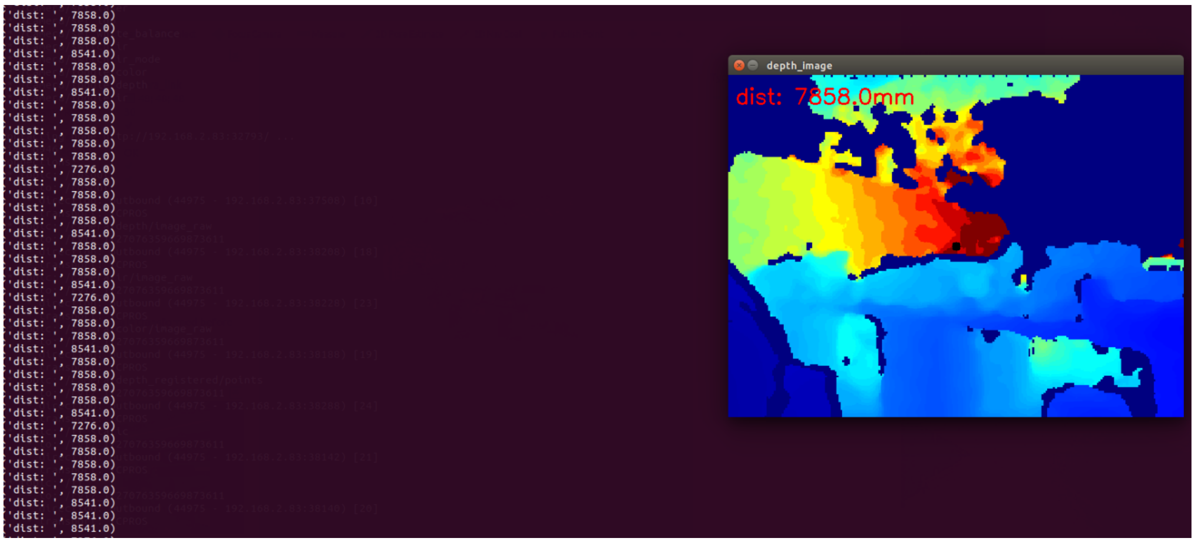
2. Startup and Operation

2.1. Startup

Enter in the terminal:

```
#Start camera
ros2 launch orbbec_camera dabai_dcw2.launch.py
#Start ranging program
ros2 run dofbot_pro_depth get_center_dis
```

After the program starts, it will get the distance of the image center point by default. If this distance is not within the effective range of the depth camera, an error may occur. Normal startup is as shown below:

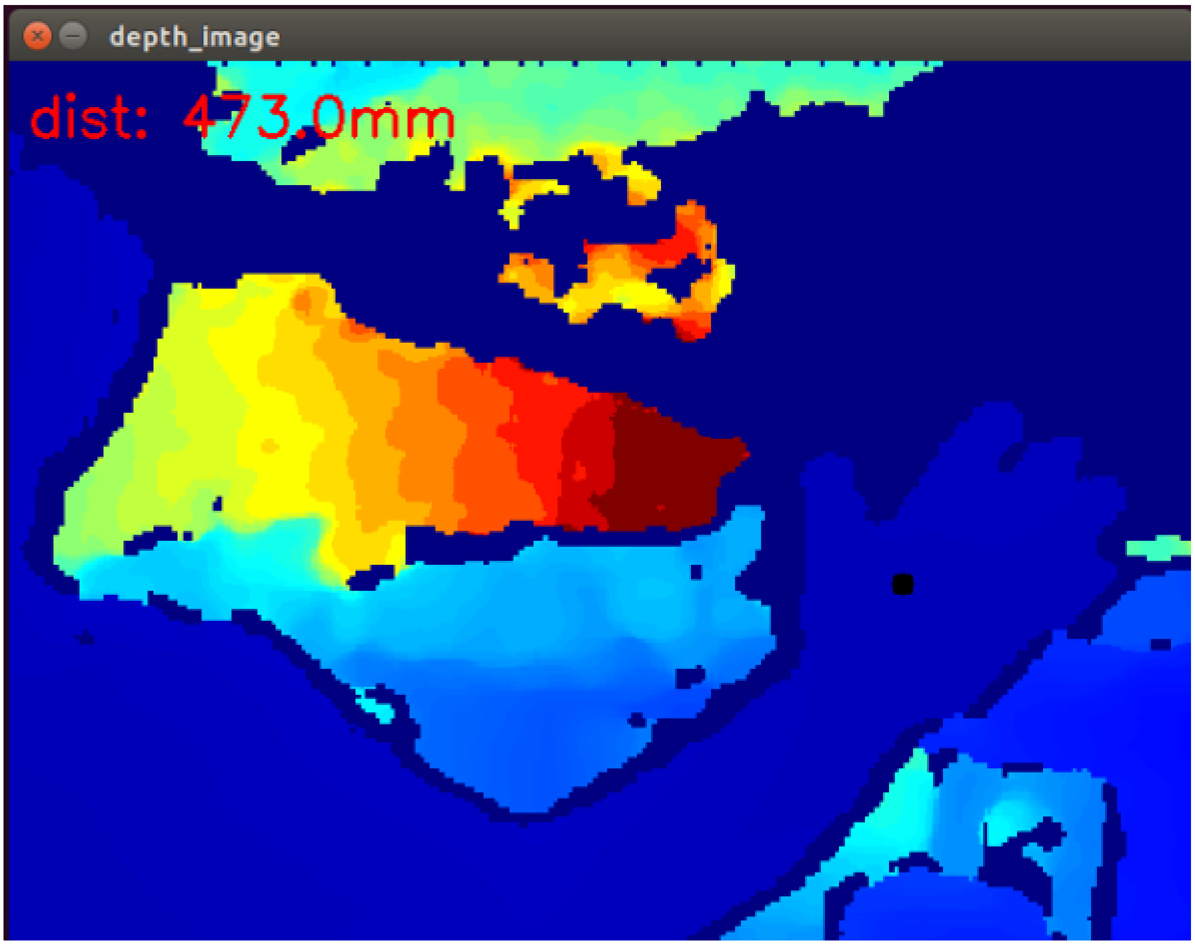


```
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7276.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7276.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 7276.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 7858.0)
'dist: ', 8541.0)
'dist: ', 8541.0)
'dist: ', 8541.0)
```

The distance of that point will be printed in the image, in millimeters (mm), and the terminal will also display the distance.

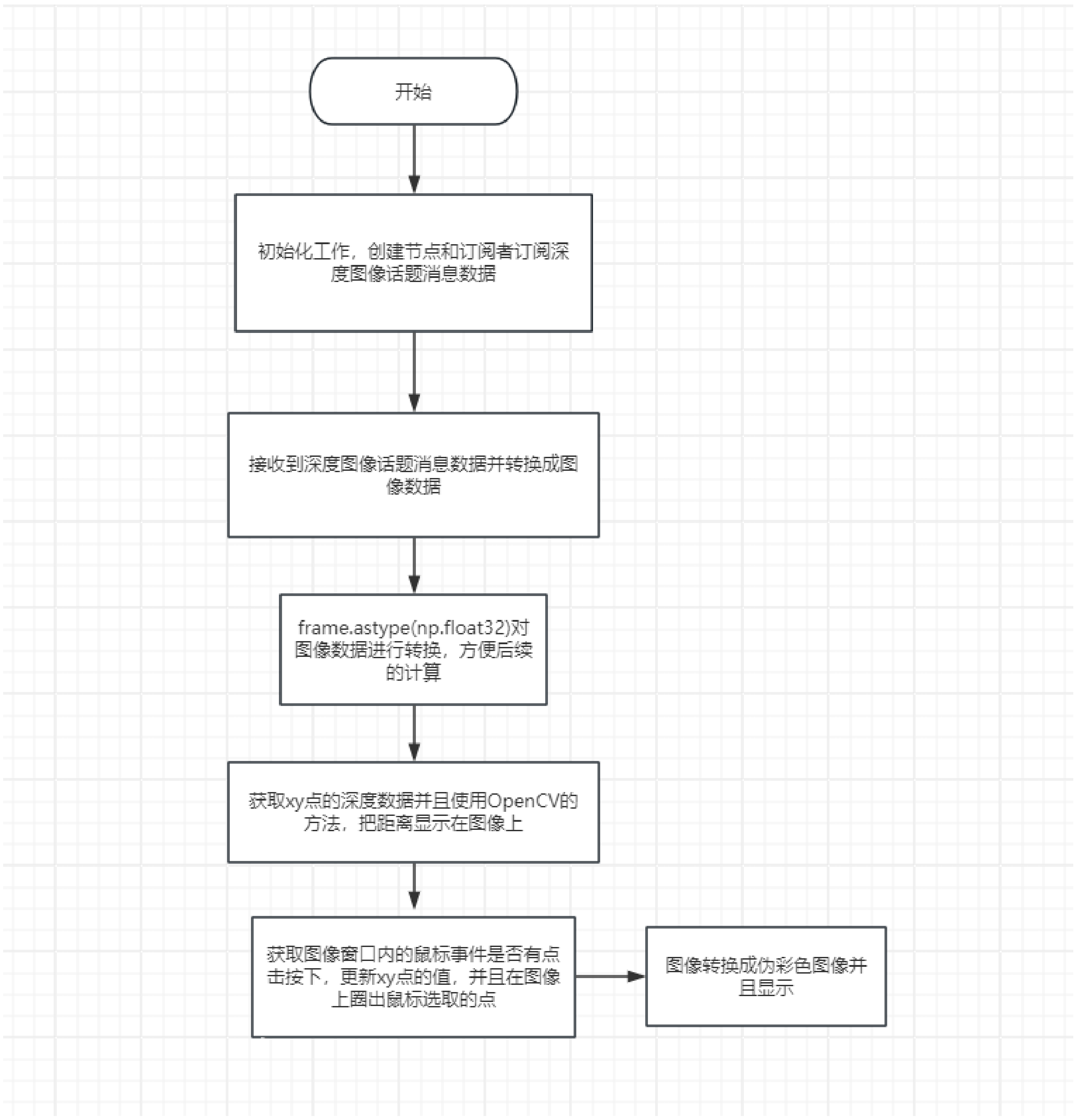
2.2. Operation

Click any point within the pseudo-color image frame with the mouse. After clicking, a black dot will appear, indicating the position of the currently selected point. The distance of the selected point will be displayed in the image, and the terminal will also display it synchronously:



Here in the demonstration, I selected a point on my hand, the distance is 47.3 centimeters, which is 473 millimeters.

3. Program Flowchart


开始
初始化工作，创建节点和订阅者订阅深度图像话题消息数据
接收到深度图像话题消息数据并转换成图像数据
frame.astype(np.float32)对图像数据进行转换，方便后续的计算
获取xy点的深度数据并且使用OpenCV的方法，把距离显示在图像上
获取图像窗口内的鼠标事件是否有点击按下，更新xy点的值，并且在图像上圈出鼠标选取的点
图像转换成伪彩色图像并且显示

4. Core Code Analysis

The code path is:

```
/home/jetson/dofbot_pro_ws/src/dofbot_pro_depth/dofbot_pro_depth/get_center_dis.
py
```

Import libraries

```
#ros2py library
import rclpy
from rclpy.node import Node
#opencv image processing library
import cv2 as cv
#cv_bridge library, used for converting message data and images
from cv_bridge import CvBridge
#import image message data type
from sensor_msgs.msg import Image
#import numpy library, for converting image data
import numpy as np
```

Define image encoding format

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

Initialize data,

```
#define a subscriber to subscribe to depth image topic messages
self.sub = rospy.Subscriber("/camera/depth/image_raw", Image, self.topic)
#define the name of the image window, convenient for subsequent programs to
recognize mouse click events
self.window_name = "depth_image"
#initial xy point coordinates are the center point, which is (320,240)
self.y = 240
self.x = 320
#create a CvBridge object, convenient for converting depth image topic message
data and image data
self.depth_bridge = CvBridge()
```

Define click_callback function, used to handle detected mouse events,

```
def click_callback(self, event, x, y, flags, params):
    '''event equals 1 means left mouse button pressed,
    cv.EVENT_MOUSEMOVE: mouse movement event, value 0
    cv.EVENT_LBUTTONDOWN: left button press event, value 1
    cv.EVENT_LBUTTONUP: left button release event, value 2
    cv.EVENT_RBUTTONDOWN: right button press event, value 3
    cv.EVENT_RBUTTONUP: right button release event, value 4
    cv.EVENT_MBUTTONDOWN: middle button press event, value 5
    cv.EVENT_MBUTTONUP: middle button release event, value 6
    cv.EVENT_LBUTTONDBLCLK: left button double click event, value 7
    cv.EVENT_RBUTTONDBLCLK: right button double click event, value 8
    cv.EVENT_MBUTTONDBLCLK: middle button double click event, value 9'''
    if event == 1:
        #After pressing the left button, x and y will be returned, update them
to the previously defined self.x and self.y
        self.x = x
        self.y = y
        print(self.x)
        print(self.y)
```

Write the subscriber's callback function to process the received depth image topic data,

```
def topic(self,msg):
    #Use the created depth_bridge's imgmsg_to_cv2 method to convert the received
msg data into image data. The input parameters are the received msg data and the
image encoding format, here the value is '32FC1'
    depth_image = self.depth_bridge.imgmsg_to_cv2(msg, encoding[1])
    #Set the image resolution to 640*480 resolution
    frame = cv.resize(depth_image, (640, 480))
    #Data conversion, convert the original uint8 data to float32 type data, to
avoid overflow and facilitate getting distance information later
    depth_image_info = frame.astype(np.float32)
    #Get the distance of that point according to xy values
    dist = depth_image_info[self.y,self.x]
    #Convert the depth image to pseudo-color image
    depth_image_orin = cv.applyColorMap(cv.convertScaleAbs(depth_image, 
alpha=0.03),    cv.COLORMAP_JET)
    print("dist: ",dist)
    #Round to 3 decimal places
    dist = round(dist,3)
    #Convert the distance information from number to string and combine with unit 
'mm'
    dist = 'dist: ' + str(dist) + 'mm'
    #Use opencv's putText function to print the distance information into the 
image
    cv.putText(depth_image_orin, dist,  (10, 40), cv.FONT_HERSHEY_SIMPLEX, 1.0, 
(0, 0, 255), 2)
    #Detect mouse events within the self.window_name window, enter click_callback 
callback function to handle mouse events
    cv.setMouseCallback(self.window_name, self.click_callback)
    #Use opencv's circle function to circle the mouse-clicked point on the image
    cv.circle(depth_image_orin,(self.x,self.y),1,(0,0,0),10)
    #Display image
    cv.imshow(self.window_name, depth_image_orin)
    cv.waitKey(1)
```