Deep Pseudo-color Image

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

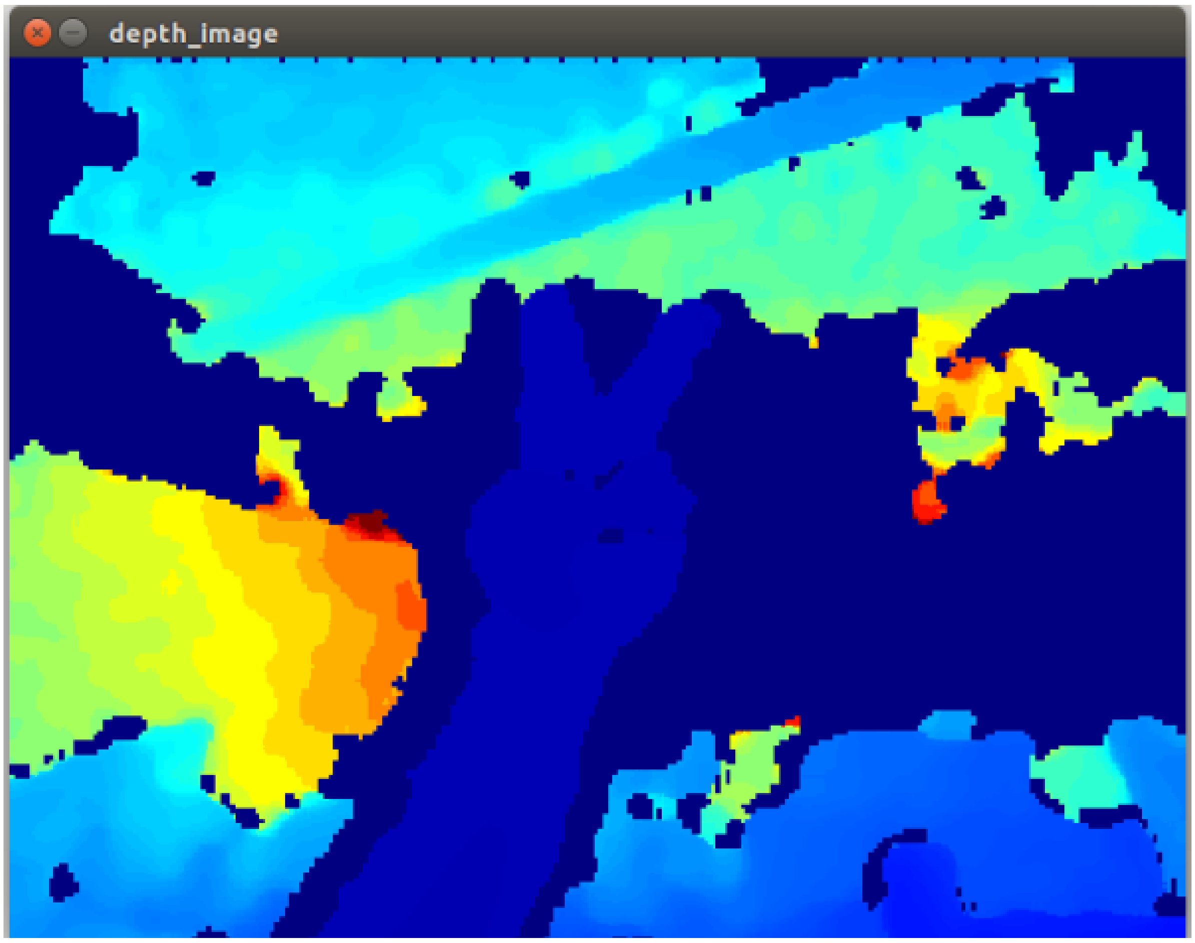
After the program starts, it will convert the subscribed black and white depth image into a pseudo-color image. According to the depth distance information, the image will display different degrees of colors.

2. Startup and Operation

Enter in the terminal:

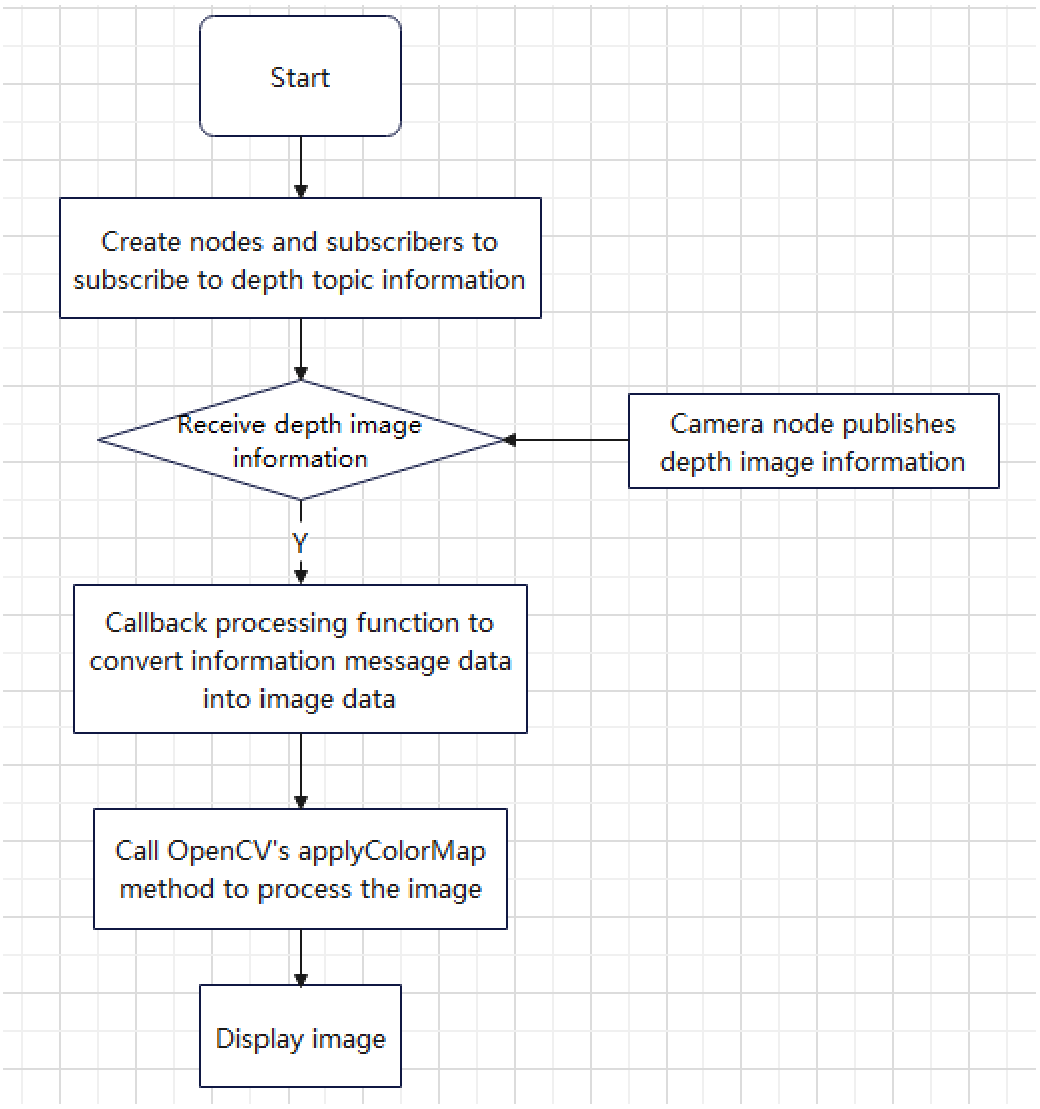
```
#Start camera
ros2 launch orbbec_camera dabai_dcw2.launch.py
#Start depth to pseudo-color image program
ros2 run dofbot_pro_depth depth_to_color
```

After successful startup, as shown in the figure below, a window will be generated displaying the pseudo-color image:



From the actual distance, the differences in depth information are intuitively reflected by colors.

3. Program Flowchart


Start
Create nodes and subscribers to subscribe to depth topic information
Receive depth image information
Camera node publishes depth image information
Y
Callback processing function to convert information message data into image data
Call OpenCV's applyColorMap method to process the image
Display image

4. Core Code Analysis

The code path is as follows:

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

Code analysis:

Import necessary 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 to image data
from cv_bridge import CvBridge
#import image data message type
from sensor_msgs.msg import Image
```

Define image encoding format

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

Create subscriber and CvBridge object

```
#define a subscriber, subscribe to /camera/depth/image_raw topic message data,
callback function is topic
self.sub = self.create_subscription(Image, '/camera/depth/image_raw',
self.topic, 10)
self.depth_bridge = CvBridge()
```

Callback function, specifically processes received image message data and converts and displays the image

```
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_orin = self.depth_bridge.imgmsg_to_cv2(msg, encoding[1])
    #Call opencv's applyColorMap method to convert the depth image obtained in
the previous step into a pseudo-color image. The input parameters are the scaled
depth image and color mapping type
    depth_to_color_image = cv.applyColorMap(cv.convertScaleAbs(depth_image_orin,
alpha=0.03), cv.COLORMAP_JET)
    #Display image
    cv.imshow(self.window_name, depth_to_color_image)
    cv.waitKey(1)
```

cv.convertScaleAbs: Used for processing images to enhance contrast or adjust brightness. The input parameters are:

- **frame**: Input image, usually a `numpy` array.
- **alpha**: Scaling factor, default is 1. It affects the scaling of pixel values.

cv.applyColorMap: Used to apply pseudo-color mapping to grayscale images, it can enhance the visual effect of images. The input parameters are:

- **frame**: Input grayscale image, usually a single-channel `numpy` array.
- **colormap**: Color mapping type. The types available here include the following options to present different effects:
    - **COLORMAP_AUTUMN**: Gradient from black to red.
    - **COLORMAP_BONE**: Gradient from gray to blue.
    - **COLORMAP_JET**: Gradient from blue to red, commonly used for heat maps.
    - **COLORMAP_WINTER**: Gradient from blue to green.
    - **COLORMAP_RAINBOW**: Rainbow color gradient.
    - **COLORMAP_OCEAN**: Gradient from dark blue to light blue.
    - **COLORMAP_SUMMER**: Gradient from green to yellow.
    - **COLORMAP_SPRING**: Gradient from pink to yellow.
    - **COLORMAP_COOL**: Gradient from cyan to pink.
    - **COLORMAP_HSV**: Gradient based on HSV color space.
    - **COLORMAP_PINK**: Gradient from black to pink.
    - **COLORMAP_HOT**: Gradient from black to red to yellow

You can change the default COLORMAP_JET in the program to the above parameters to test and see different effects.