Face detection

1. Content Description

This course implements color image acquisition and face detection using the mediapipe framework. This section requires entering commands in the terminal. The terminal you open depends on your board type. This section 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 **[01. Robot Configuration and Operation Guide] --[5.Enter Docker (For JETSON Nano and RPi 5)]**. For Orin boards, simply open the terminal and enter the commands mentioned in this section.

2. Program Startup

For the Raspberry Pi 5 controller, you must first enter the Docker container. For the Orin controller, this is not necessary.

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

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

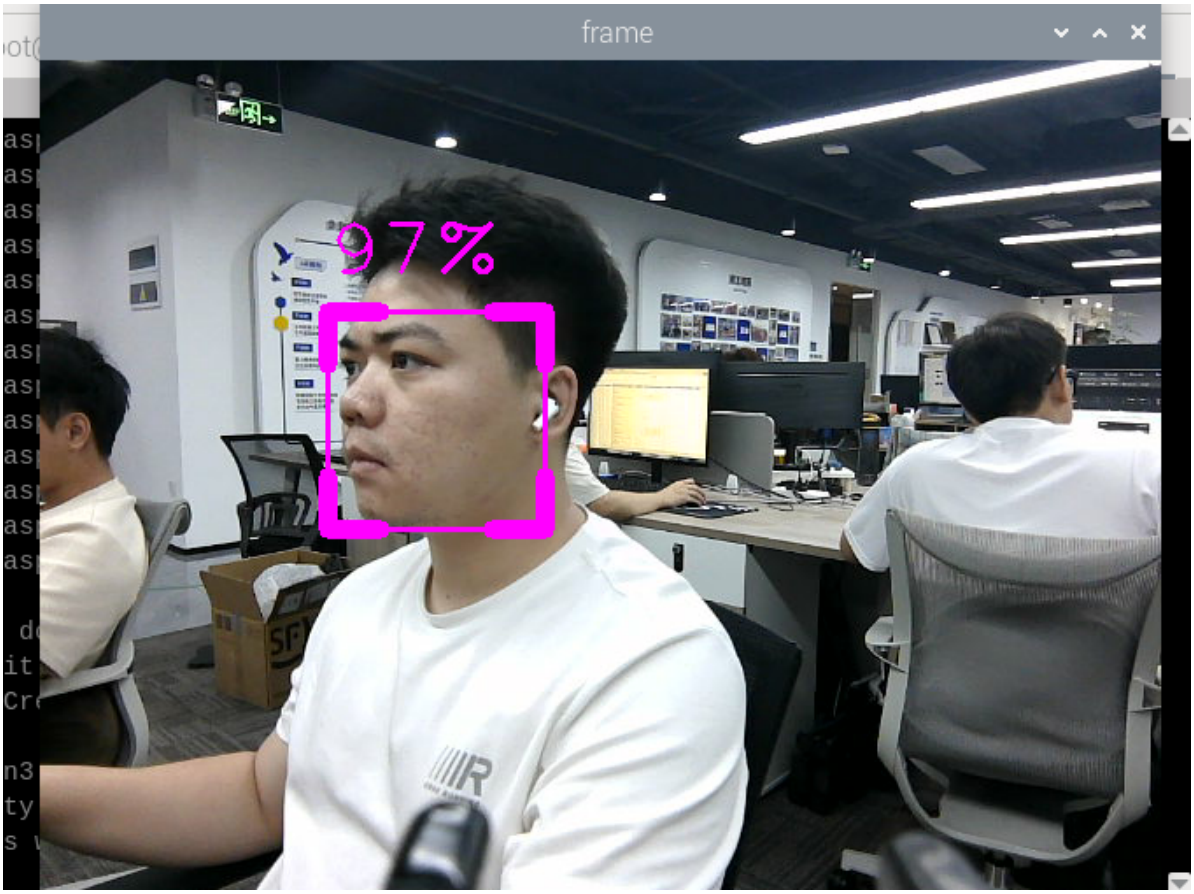
First, enter the following command in the terminal to start the camera.

```
#usb camera
ros2 launch usb_cam camera.launch.py
#nuwa camera
ros2 launch ascamera hp60c.launch.py
```

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

```
ros2 run yahboomcar_mediapipe 05_FaceDetection
```

After running the program, as shown in the image below, detected faces are outlined and the detection score is displayed. A higher score indicates better face recognition.



3. Core Code Analysis

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_mediapipe/yahboomcar_mediapipe/07_FaceDetection.py`

- Orin motherboard

The program code path is `/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/07_FaceDetection.py`

Import the required library files.

```
import time
import rclpy
from rclpy.node import Node
import cv2 as cv
import numpy as np
import mediapipe as mp
from geometry_msgs.msg import Point
from yahboomcar_msgs.msg import PointArray
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
import os
```

Initialize data and define publishers and subscribers,

```
def __init__(self, minDetectionCon=0.5):
    super().__init__('face_detector')
    #Use the class in the mediapipe library to define a face detection object
    self.mpFaceDetection = mp.solutions.face_detection
    self.mpDraw = mp.solutions.drawing_utils
    self.facedetection =
self.mpFaceDetection.FaceDetection(min_detection_confidence=minDetectionCon)
    self.bridge = CvBridge()
    #Define subscribers for the color image topic
    camera_type = os.getenv('CAMERA_TYPE', 'usb')
    topic_name = '/ascamera_hp60c/camera_publisher/rgb0/image' if camera_type ==
'nuwa' else '/usb_cam/image_raw'
    self.subscription = self.create_subscription(
    Image,
    topic_name,
    self.image_callback,
    10)
```

Color image callback function,

```
def image_callback(self, msg):
    #Use CvBridge to convert color image message data into image data
    frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    #Pass the obtained image into the defined findFaces function to perform face
detection program
    frame, _ = self.findFaces(frame)
    cv.imshow('Face Detection', frame)
```

findFaces function,

```
def findFaces(self, frame):
    #Convert the color space of the incoming image from BGR to RGB to facilitate
subsequent image processing
    img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    #Call the process function in the mediapipe library for image processing.
During init, the self.facedetection object is created and initialized.
    self.results = self.facedetection.process(img_RGB)
    bboxs = []
    #Determine whether self.results.detections exists, that is, whether a face is
recognized
    if self.results.detections:
        #Traverse the id and detection data
        for id, detection in enumerate(self.results.detections):
            #Get bounding box coordinates
            bboxC = detection.location_data.relative_bounding_box
            ih, iw, ic = frame.shape
            bbox = int(bboxC.xmin * iw), int(bboxC.ymin * ih), \
            int(bboxC.width * iw), int(bboxC.height * ih)
            #Storing test results
            bboxs.append([id, bbox, detection.score])
            #Call the fancyDraw function to draw the bounding box
            frame = self.fancyDraw(frame, bbox)
            #Display the face recognition score on the image
            cv.putText(frame, f'{int(detection.score[0] * 100)}%',(bbox[0],
bbox[1] - 20), cv.FONT_HERSHEY_PLAIN,3, (255, 0, 255), 2)
    return frame, bboxs
```

fancyDraw function draws the bounding box according to the value of the detection result bbox

```
def fancyDraw(self, frame, bbox, l=30, t=10):
    x, y, w, h = bbox
    x1, y1 = x + w, y + h
    cv.rectangle(frame, (x, y),(x + w, y + h), (255, 0, 255), 2)
    # Top left x,y
    cv.line(frame, (x, y), (x + l, y), (255, 0, 255), t)
    cv.line(frame, (x, y), (x, y + l), (255, 0, 255), t)
    # Top right x1,y
    cv.line(frame, (x1, y), (x1 - l, y), (255, 0, 255), t)
    cv.line(frame, (x1, y), (x1, y + l), (255, 0, 255), t)
    # Bottom left x1,y1
    cv.line(frame, (x, y1), (x + l, y1), (255, 0, 255), t)
    cv.line(frame, (x, y1), (x, y1 - l), (255, 0, 255), t)
    # Bottom right x1,y1
    cv.line(frame, (x1, y1), (x1 - l, y1), (255, 0, 255), t)
    cv.line(frame, (x1, y1), (x1, y1 - l), (255, 0, 255), t)
    return frame
```