Face special effects

1. Content Description

This program implements color image acquisition and face detection and special effects using the dlib library. This section requires entering commands in the terminal.

The terminal you open depends on your motherboard 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 the Orin board, simply open a 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; the Orin controller does not need to.

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]).**

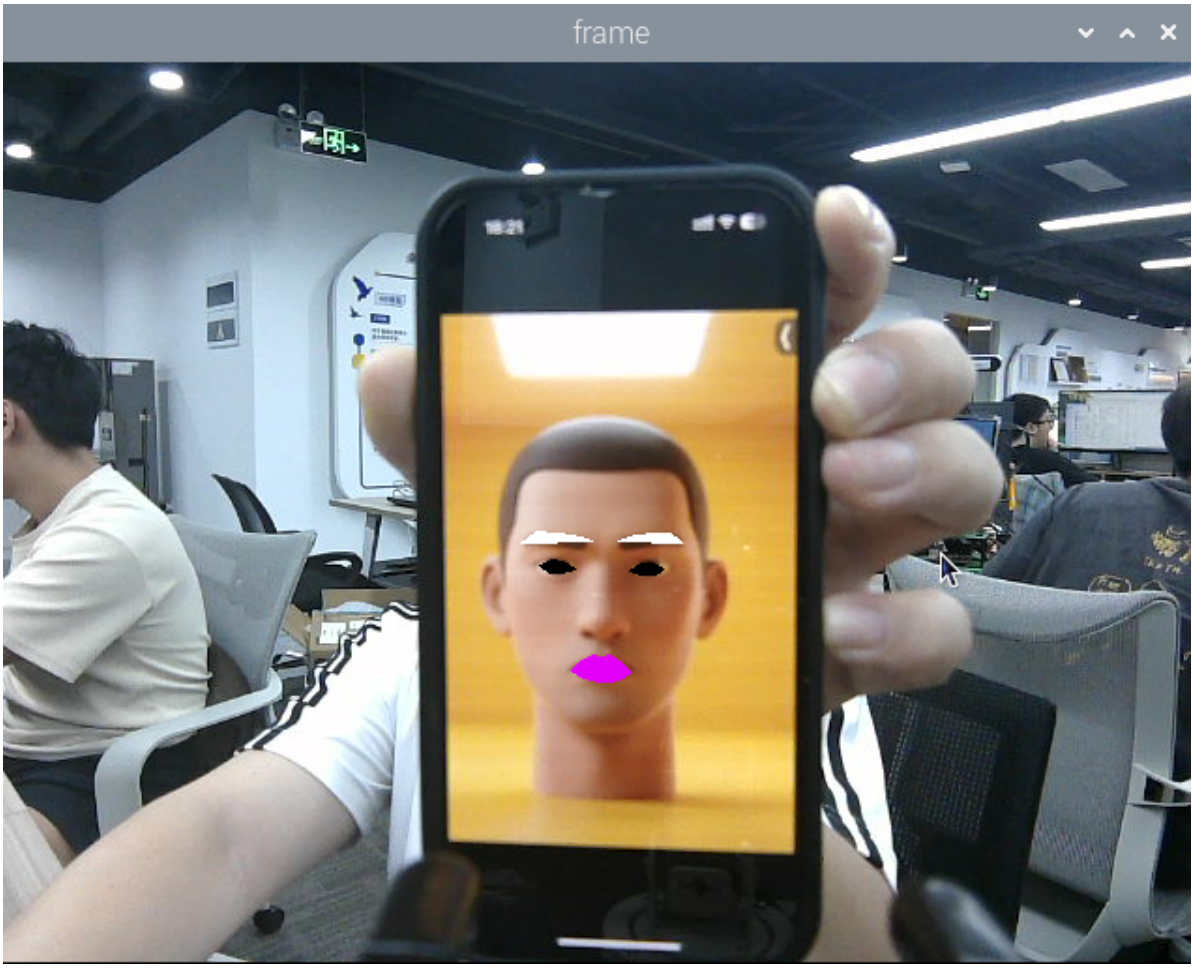
First, enter the following command in the terminal to start the camera:

```
#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 effects program:

```
ros2 run yahboomcar_mediapipe 06_FaceLandmarks
```

After running the program, as shown in the image below, it first detects the face and then applies special effects to the eyebrows, eyes, and mouth.



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/06_FaceLandmarks.py`

- Orin motherboard

The program code path is `/home/jetson/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_mediapipe/06_FaceLandmarks.py`

Import the library files used.

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

About the dlib library:

dlib is a modern C++ toolkit that includes machine learning algorithms and tools for building complex software in C++ to solve real-world problems. It is widely used in industry and academia for applications in robotics, embedded devices, mobile phones, and large-scale high-performance computing environments. The dlib library uses 68 points to mark key facial features, such as points 18-22 for the right eyebrow and points 51-68 for the mouth. Faces are detected using the get_frontal_face_detector module in the dlib library, and facial feature values are predicted using the shape_predictor_68_face_landmarks.dat feature data. ,

The 68 facial landmarks in dlib are arranged in the following order:

- 0-16: Chinline
- 17-21: Right eyebrow
- 22-26: Left eyebrow
- 27-35: Nose bridge and tip
- 36-41: Right eye
- 42-47: Left eye
- 48-67: Lip outline

Initialize the data and define the publisher and subscriber ,

```
def __init__(self, dat_file):
    super().__init__('face_landmarks')
    #Import database
    self.dlib_facelandmark = dlib.shape_predictor(dat_file)
    #Define face detection objects
    self.hog_face_detector = dlib.get_frontal_face_detector()
    self.dlib_facelandmark = dlib.shape_predictor(self.dat_file)

    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)
```

The topic of color images returns to the function,

```
def image_callback(self, msg):
    frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
    #Pass the obtained image into the defined get_face function and return the
detected image
    frame = self.get_face(rgb_image, draw=False)
    #Call the prettify_face function to perform special effects processing on the
image
    frame = self.prettify_face(frame, eye=True, lips=True, eyebrow=True,
draw=True)
    cv.imshow('Face Landmarks', frame)
```

get_face function, detects faces,

```
def get_face(self, frame, draw=True):
    #Convert the image space and convert bgr into grayscale to facilitate
subsequent image processing
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    #Input grayscale image and detect face
    self.faces = self.hog_face_detector(gray)
    for face in self.faces:
        self.face_landmarks = self.dlib_facelandmark(gray, face)
        if draw:
            for n in range(68):
                x = self.face_landmarks.part(n).x
                y = self.face_landmarks.part(n).y
                cv.circle(frame, (x, y), 2, (0, 255, 255), 2)
                cv.putText(frame, str(n), (x, y), cv.FONT_HERSHEY_SIMPLEX, 0.6,
(0, 255, 255), 2)
    return frame
```

prettify_face function, adds special effects to the face,

```
def prettify_face(self, frame, eye=True, lips=True, eyebrow=True, draw=True):
    #Eye
    if eye:
        leftEye = self.get_lmList(frame, 36, 42)
        rightEye = self.get_lmList(frame, 42, 48)
        if draw:
            if len(leftEye) != 0: frame = cv.fillConvexPoly(frame,
np.mat(leftEye), (0, 0, 0))
            if len(rightEye) != 0: frame = cv.fillConvexPoly(frame,
np.mat(rightEye), (0, 0, 0))
    #lips
    if lips:
        lipIndexlistA = [51, 52, 53, 54, 64, 63, 62]
        lipIndexlistB = [48, 49, 50, 51, 62, 61, 60]
        lipsUpA = self.get_lipList(frame, lipIndexlistA, draw=True)
        lipsUpB = self.get_lipList(frame, lipIndexlistB, draw=True)
        lipIndexlistA = [57, 58, 59, 48, 67, 66]
        lipIndexlistB = [54, 55, 56, 57, 66, 65, 64]
        lipsDownA = self.get_lipList(frame, lipIndexlistA, draw=True)
        lipsDownB = self.get_lipList(frame, lipIndexlistB, draw=True)
        if draw:
            if len(lipsUpA) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsUpA), (249, 0, 226))
            if len(lipsUpB) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsUpB), (249, 0, 226))
            if len(lipsDownA) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsDownA), (249, 0, 226))
            if len(lipsDownB) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsDownB), (249, 0, 226))
    #Eyebrow
    if eyebrow:
        lefteyebrow = self.get_lmList(frame, 17, 22)
        righteyebrow = self.get_lmList(frame, 22, 27)
        if draw:
            if len(lefteyebrow) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lefteyebrow), (255, 255, 255))
            if len(righteyebrow) != 0: frame = cv.fillConvexPoly(frame,
np.mat(righteyebrow), (255, 255, 255))
```

```
    return frame
```

get_lmList gets the facial coordinate function,

```
def get_lmList(self, frame, p1, p2, draw=True):
    #Define an empty list
    lmList = []
    # Determine whether a face is detected
    if len(self.faces) != 0:
        #Traverse the face list and get the xy coordinates of each point in the
interval
        for n in range(p1, p2):
            x = self.face_landmarks.part(n).x
            y = self.face_landmarks.part(n).y
            #Add the coordinates of the points on the face to the list
            lmList.append([x, y])
            if draw:
                next_point = n + 1
                if n == p2 - 1: next_point = p1
                x2 = self.face_landmarks.part(next_point).x
                y2 = self.face_landmarks.part(next_point).y
                cv.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)
    return lmList
```