5. Face recognition

5.1. Introduction

MediaPipe is an open-source data stream processing machine learning application development framework developed by Google. It is a graph-based data processing pipeline used to build data sources in various forms, such as video, audio, sensor data, and any time series data. MediaPipe is cross-platform and can run on embedded platforms (such as Raspberry Pi), mobile devices (iOS and Android), workstations and servers, and supports mobile GPU acceleration. MediaPipe provides cross-platform, customizable ML solutions for real-time and streaming media. The core framework of MediaPipe is implemented in C++ and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include packets, streams, calculators, graphs, and subgraphs.

Features of MediaPipe:

- End-to-end acceleration: built-in fast ML inference and processing can be accelerated even on ordinary hardware.
- Build once, deploy anywhere: unified solution for Android, iOS, desktop/cloud, web, and IoT.
- Ready-to-use solution: cutting-edge ML solution that showcases the full capabilities of the framework.
- Free and open source: framework and solution under Apache 2.0, fully extensible and customizable.

5.2, Face recognition

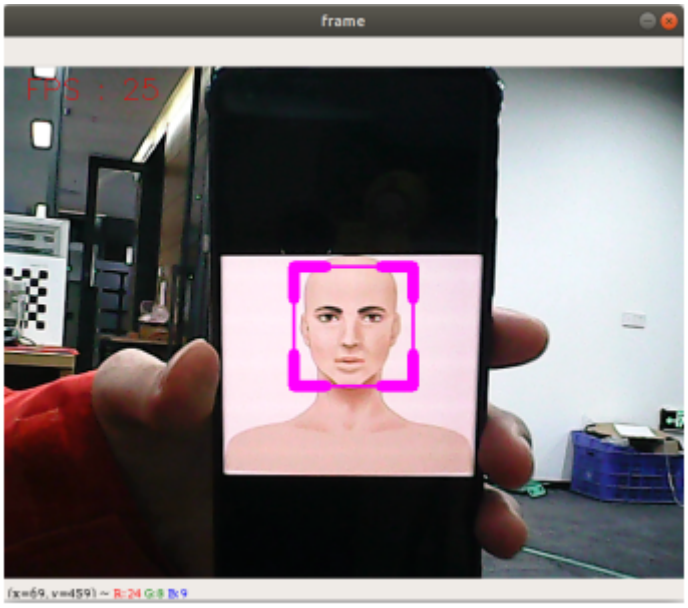
5.2.1, Start

Start the camera

```
roslaunch ascamera hp60c.launch
```

Terminal input,

```
roslaunch yahboomcar_mediapipe 05_FaceEyeDetection.launch
```



5.2.2, Source code

Source code location:
/home/yahboom/ascam_ws/src/yahboomcar_mediapipe/scripts/05_FaceEyeDetection.py

```
#!/usr/bin/env python3
# encoding: utf-8
import time
import rospy
import rospkg
import cv2 as cv
from cv_bridge import CvBridge
from sensor_msgs.msg import CompressedImage, Image


class FaceEyeDetection:
    def __init__(self):
        self.bridge = CvBridge()
        rospy.on_shutdown(self.cancel)
        rospy.init_node("FaceEyeDetection", anonymous=False)
        self.eyeDetect = cv.CascadeClassifier(
            rospkg.RosPack().get_path("yahboomcar_mediapipe") +
"/scripts/file/haarcascade_eye.xml")
        self.faceDetect = cv.CascadeClassifier(
            rospkg.RosPack().get_path("yahboomcar_mediapipe") +
"/scripts/file/haarcascade_frontalface_default.xml")
        self.pub_rgb = rospy.Publisher("/FaceEyeDetection/image", Image,
queue_size=1)
        # Subscribe to the /ascamera_hp60c/rgb0/image topic
        self.sub_image = rospy.Subscriber('/ascamera_hp60c/rgb0/image', Image,
self.image_callback, queue_size=1)
        self.content_index = 0
        self.pTime = 0
        self.content = ["face", "eye", "face_eye"]


    def cancel(self):
        self.pub_rgb.unregister()
```

```
    def face(self, frame):
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        faces = self.faceDetect.detectMultiScale(gray, 1.3)
        for face in faces: frame = self.faceDraw(frame, face)
        return frame

    def eye(self, frame):
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        eyes = self.eyeDetect.detectMultiScale(gray, 1.3)
        for eye in eyes:
            cv.circle(frame, (int(eye[0] + eye[2] / 2), int(eye[1] + eye[3] /
2)), (int(eye[3] / 2)), (0, 0, 255), 2)
        return frame

    def faceDraw(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

    def pub_img(self, frame):
        self.pub_rgb.publish(self.bridge.cv2_to_imgmsg(frame, "bgr8"))

    def image_callback(self, msg):
        try:
            # Convert the ROS Image message to an OpenCV image
            frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding="bgr8")

            # Process the image based on the current mode
            if self.content[self.content_index] == "face":
                frame = self.face(frame)
            elif self.content[self.content_index] == "eye":
                frame = self.eye(frame)
            else:  # face_eye
                frame = self.eye(self.face(frame))

            # Calculate and display FPS
            cTime = time.time()
            fps = 1 / (cTime - self.pTime)
            self.pTime = cTime
            text = "FPS: {}".format(int(fps))
```

```
            cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0,
0, 255), 1)

            # Display the image
            cv.imshow('frame', frame)

            # Publish the processed image
            self.pub_img(frame)

            # Check for key press to change mode or quit
            action = cv.waitKey(1) & 0xFF
            if action == ord('f') or action == ord('F'):
                self.content_index = (self.content_index + 1) %
len(self.content)
            elif action == ord('q') or action == ord('Q'):
                rospy.signal_shutdown("User requested shutdown")
        except Exception as e:
            rospy.logerr("Could not process image: %s" % e)


if __name__ == '__main__':
    face_eye_detection = FaceEyeDetection()
    rospy.spin()
    cv.destroyAllWindows()
```