7.Face detection

7.1. 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 for building and using multiple forms of data sources, such as video, audio, sensor data, and any time series data. MediaPipe is cross-platform and can run on embedded platforms (Raspberry Pi, etc.), 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 Packet, Stream, Calculator, Graph and Subgraph.

MediaPipe Features:

- End-to-end acceleration: Built-in fast ML inference and processing accelerates even on commodity hardware.
- Build once, deploy anywhere: Unified solution for Android, iOS, desktop/cloud, web and IoT.
- Ready-to-use solutions: cutting-edge ML solutions that showcase the full capabilities of the framework.
- Free and open source: frameworks and solutions under Apache2.0, fully extensible and customizable.

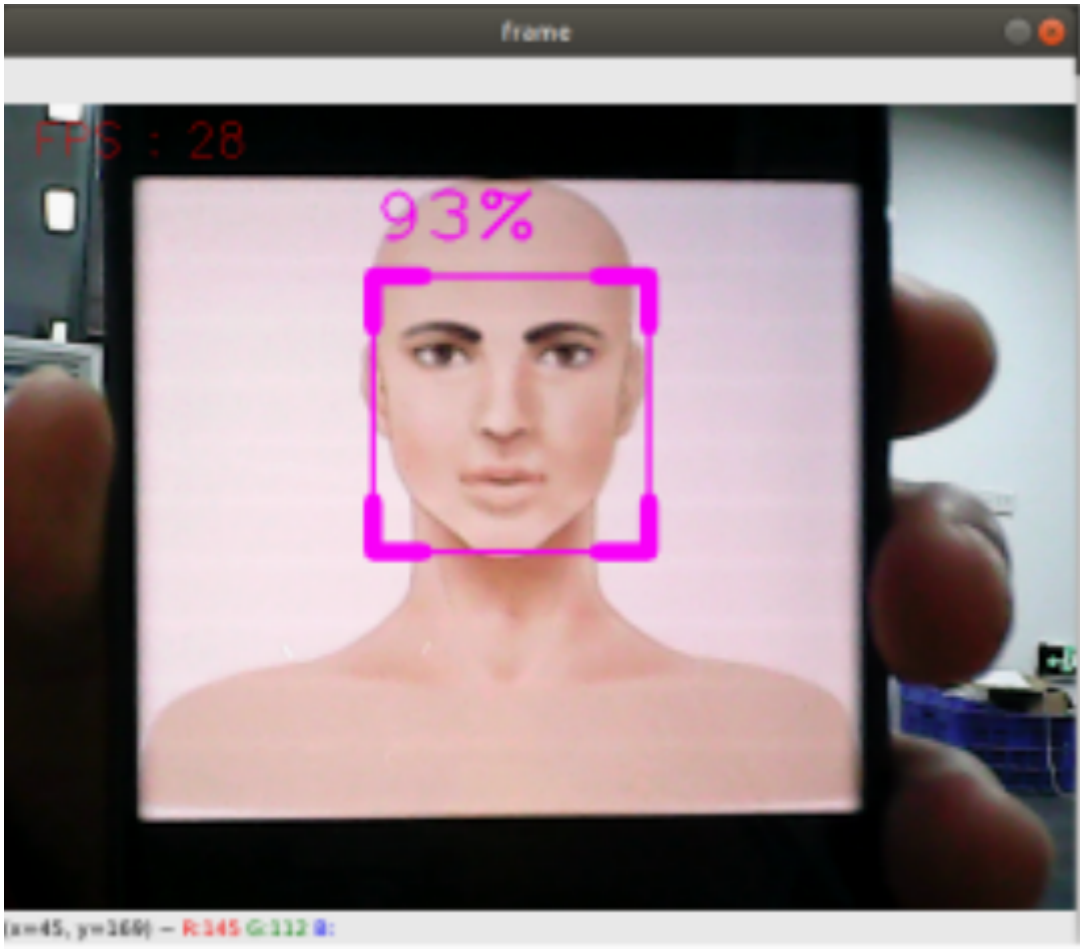
7.2. Face detection

7.2.1.Startup

Note: Before running this case, please make sure that the [/dev/video0] device has been successfully mounted into the docker container, otherwise the camera will not be opened.

Enter the docker container and execute:

```
cd
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_media
pipe
#Face detection
python3 07_FaceDetection.py
```



7.2.2.Source code

Source code location:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_mediapipe/yahboomcar_media
pipe/07_FaceDetection.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import mediapipe as mp
import cv2 as cv
import time


class FaceDetector:
    def __init__(self, minDetectionCon=0.5):
        self.mpFaceDetection = mp.solutions.face_detection
        self.mpDraw = mp.solutions.drawing_utils
        self.facedetection =
self.mpFaceDetection.FaceDetection(min_detection_confidence=minDetectionCon)

    def findFaces(self, frame):
        img_RGB = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
        self.results = self.facedetection.process(img_RGB)
        bboxs = []
        if self.results.detections:
            for id, detection in enumerate(self.results.detections):
```

```
                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)
                bboxs.append([id, bbox, detection.score])
                frame = self.fancyDraw(frame, bbox)
                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

    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


if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    capture.set(6, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    print("capture get FPS : ", capture.get(cv.CAP_PROP_FPS))
    pTime, cTime = 0, 0
    face_detector = FaceDetector(0.75)
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv.flip(frame, 1)
        frame,_ = face_detector.findFaces(frame)
        if cv.waitKey(1) & 0xFF == ord('q'): break
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        text = "FPS : " + str(int(fps))
        cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,
255), 1)
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```