12. Face Detection

12.1. 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 Apache2.0, fully extensible and customizable.

12.2. Face Detection

Source code location: /home/pi/project_demo/07.AI_Visual_Recognition/mediapipe/12.Face_detection/12_Face_detection.ipynb

```
#!/usr/bin/env python3
# encoding: utf-8
#导入相关的模块 Import related modules
import mediapipe as mp
import threading
import cv2 as cv
import time
import math
from time import sleep
import ipywidgets.widgets as widgets
image_widget = widgets.Image(format='jpeg', width=640, height=480)  #设置摄像头显示组件 Set up the camera display component
```

```python
# 将BGR图像转换为JPEG格式的字节流 Convert a BGR image to a JPEG byte stream
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv.imencode('.jpg', value)[1])
```

```python
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
```

```python
if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    # capture.set(0, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 320)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 240)
    print("capture get FPS : ", capture.get(cv.CAP_PROP_FPS))
    pTime, cTime = 0, 0
```

```
    face_detector = FaceDetector(0.75)
    display(image_widget)
    try:
        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, f"FPS: {fps:.1f}", (10, 30),
cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
            image_widget.value = bgr8_to_jpeg(frame)
            # cv.imshow('frame', frame)
    except KeyboardInterrupt:
        capture.release()
        # cv.destroyAllWindows()
```

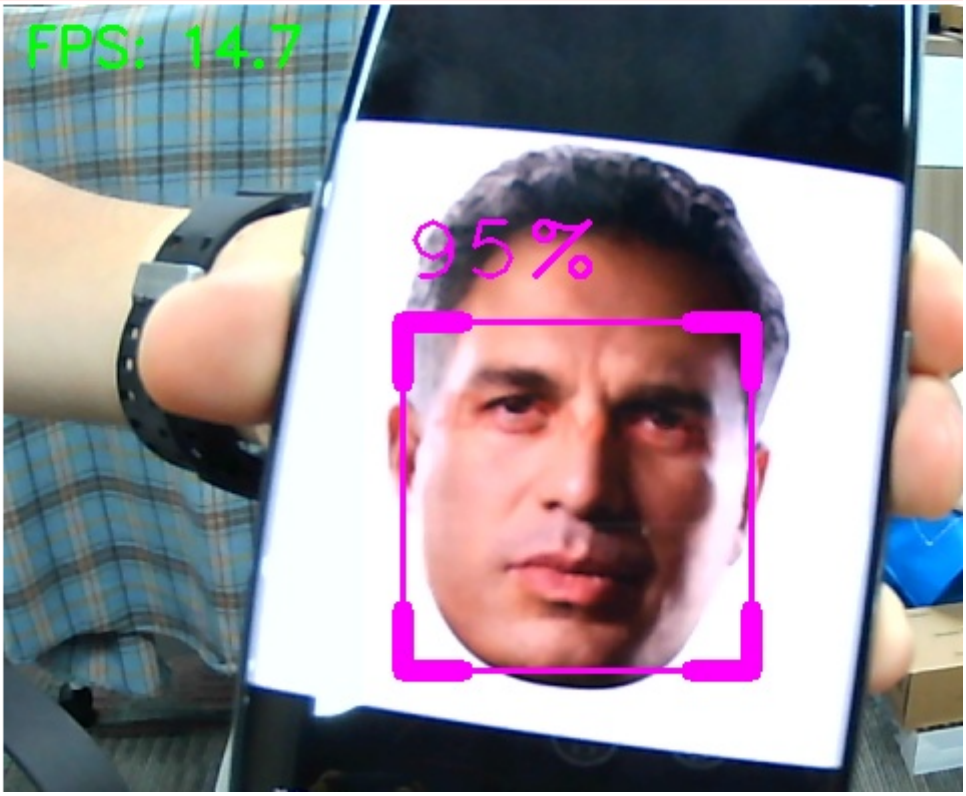
12_Face_detection.ipynb

Code

```
capture.release()
# cv.destroyAllWindows()
```

```
capture get FPS :  15.0
```

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called a
W0000 00:00:1722592147.242512  867797 inference_feedback_manager.c
ling support for feedback tensors.
```



```
#使用完成对象记住释放掉对象，不然下一个程序使用这个对象模块会被占用，导致无法使用
#Release resources
capture.release()
```

You need to stop the program before releasing the object