13. Face effects

13. Face effects
    13.1. Introduction
    13.2. Face Effects

13.1. 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.

13.2. Face Effects

Source code location:

/home/pi/project_demo/07.AI_Visual_Recognition/mediapipe/13.Face_effects/13_Face_effects.ipynb

```
#!/usr/bin/env python3
# encoding: utf-8
#导入相关的模块 Import related modules
import threading
import numpy as np
import cv2
import dlib
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
```

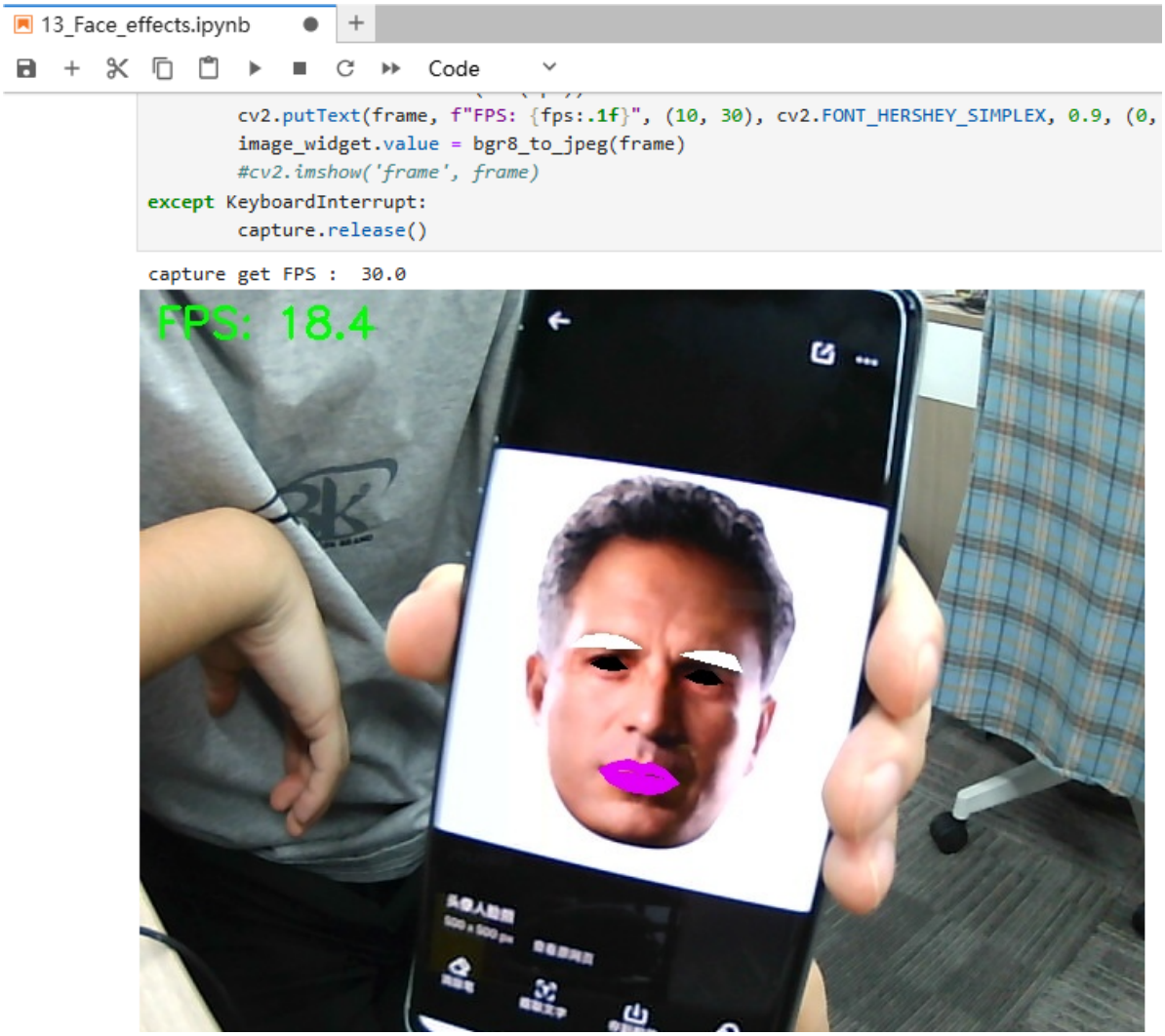
```
# 将BGR图像转换为JPEG格式的字节流 Convert a BGR image to a JPEG byte stream
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
class FaceLandmarks:
    def __init__(self, dat_file):
        self.hog_face_detector = dlib.get_frontal_face_detector()
        self.dlib_facelandmark = dlib.shape_predictor(dat_file)

    def get_face(self, frame, draw=True):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        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
                    cv2.circle(frame, (x, y), 2, (0, 255, 255), 2)
                    cv2.putText(frame, str(n), (x, y), cv2.FONT_HERSHEY_SIMPLEX,
0.6, (0, 255, 255), 2)
        return frame

    def get_lmList(self, frame, p1, p2, draw=True):
        lmList = []
        if len(self.faces) != 0:
            for n in range(p1, p2):
                x = self.face_landmarks.part(n).x
                y = self.face_landmarks.part(n).y
                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
                    cv2.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)
        return lmList

    def get_lipList(self, frame, lipIndexlist, draw=True):
        lmList = []
        if len(self.faces) != 0:
            for n in range(len(lipIndexlist)):
                x = self.face_landmarks.part(lipIndexlist[n]).x
                y = self.face_landmarks.part(lipIndexlist[n]).y
                lmList.append([x, y])
                if draw:
                    next_point = n + 1
                    if n == len(lipIndexlist) - 1: next_point = 0
                    x2 = self.face_landmarks.part(lipIndexlist[next_point]).x
                    y2 = self.face_landmarks.part(lipIndexlist[next_point]).y
                    cv2.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)
        return lmList
```

```
    def prettify_face(self, frame, eye=True, lips=True, eyebrow=True,
draw=True):
        if eye:
            leftEye = landmarks.get_lmList(frame, 36, 42)
            rightEye = landmarks.get_lmList(frame, 42, 48)
            if draw:
                if len(leftEye) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(leftEye), (0, 0, 0))
                if len(rightEye) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(rightEye), (0, 0, 0))
        if lips:
            lipIndexlistA = [51, 52, 53, 54, 64, 63, 62]
            lipIndexlistB = [48, 49, 50, 51, 62, 61, 60]
            lipsUpA = landmarks.get_lipList(frame, lipIndexlistA, draw=True)
            lipsUpB = landmarks.get_lipList(frame, lipIndexlistB, draw=True)
            lipIndexlistA = [57, 58, 59, 48, 67, 66]
            lipIndexlistB = [54, 55, 56, 57, 66, 65, 64]
            lipsDownA = landmarks.get_lipList(frame, lipIndexlistA, draw=True)
            lipsDownB = landmarks.get_lipList(frame, lipIndexlistB, draw=True)
            if draw:
                if len(lipsUpA) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(lipsUpA), (249, 0, 226))
                if len(lipsUpB) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(lipsUpB), (249, 0, 226))
                if len(lipsDownA) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(lipsDownA), (249, 0, 226))
                if len(lipsDownB) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(lipsDownB), (249, 0, 226))
        if eyebrow:
            lefteyebrow = landmarks.get_lmList(frame, 17, 22)
            righteyebrow = landmarks.get_lmList(frame, 22, 27)
            if draw:
                if len(lefteyebrow) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(lefteyebrow), (255, 255, 255))
                if len(righteyebrow) != 0: frame = cv2.fillConvexPoly(frame,
np.mat(righteyebrow), (255, 255, 255))
        return frame
```

```
if __name__ == '__main__':
    capture = cv2.VideoCapture(0)
    capture.set(6, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
    print("capture get FPS : ", capture.get(cv2.CAP_PROP_FPS))
    pTime, cTime = 0, 0
    dat_file = "./file/shape_predictor_68_face_landmarks.dat"
    landmarks = FaceLandmarks(dat_file)
    display(image_widget)
try:
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv2.flip(frame, 1)
        frame = landmarks.get_face(frame, draw=False)
        frame = landmarks.prettify_face(frame, eye=True, lips=True,
eyebrow=True, draw=True)
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'): break
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        text = "FPS : " + str(int(fps))
        cv2.putText(frame, f"FPS: {fps:.1f}", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
        image_widget.value = bgr8_to_jpeg(frame)
        #cv2.imshow('frame', frame)
except KeyboardInterrupt:
        capture.release()
```




```
#Release resources
capture.release()                    #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
```

Before releasing resources, you need to stop the program first

13_Face_effects.ipynb

Code

[*]:
```
if __name__ == '__main__':
    capture = cv2.VideoCapture(0)
    capture.set(6, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
    capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
    print("capture get FPS : ", capture.get(cv2.CAP_PROP_FPS))
    pTime, cTime = 0, 0
    dat_file = "./file/shape_predictor_68_face_landmarks.dat"
    landmarks = FaceLandmarks(dat_file)
    display(image_widget)
try:
    while capture.isOpened():
        ret, frame = capture.read()
        # frame = cv2.flip(frame, 1)
        frame = landmarks.get_face(frame, draw=False)
        frame = landmarks.prettify_face(frame, eye=True, lips=True, eyebrow=True, draw=True)
        if cv2.waitKey(1) & 0xFF == ord('q'): break
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        text = "FPS : " + str(int(fps))
        cv2.putText(frame, f"FPS: {fps:.1f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
        image_widget.value = bgr8_to_jpeg(frame)
        #cv2.imshow('frame', frame)
except KeyboardInterrupt:
        capture.release()
```

```
capture get FPS :  30.0
```