会看眼色的电影助手

队伍名称：事情不太队

队伍成员：李培伟 彭佳颖 唐芷怡

指导老师：鲍杨

内容目录

问题和主要任务描述

0.1 项目背景

我们的项目灵感主要来源于两点。第一点，课堂上老师演示的面部识别demo和去年同学做的体感2048程序项目，让我们对和人脸识别相关的内容产生了较大兴趣。第二点，最终选定的项目主体来源于现实需求，我们都是喜欢用电脑看影视剧和综艺等等的年轻人，经常会拿着薯片等各种零食边吃边看，而这就产生了一个问题，吃东西时手会弄脏，每次想要对视频进行暂停、快进或后退等操作时，都需要擦干净手再动，因此，我们希望可以解放双手、做到仅通过某些面部动作便能操作视频播放。

0.2 本项目中我们的主要任务与分工

0.2.1 编程任务

1. 获取视频流并进行面部参数识别

通过电脑摄像头实时获取含有用户面部的画面，使用68landmark对人脸进行描点，获得面部特征坐标。

2. 确定目标动作，构造识别该动作的判定方法

我们选用眨眼作为目标动作，使用面部特征坐标构造双眼特征值函数，希望通过比较算出的值和我们提前设定的标准值（阈值）来判定“双眼同时闭上再睁开（并获取时长）”和“单闭左眼或右眼（并获取时长）”的动作。

3. 对电脑内正在播放的视频进行暂停等操作

实现以下对应操作：

双眼同时闭上2秒再睁开：暂停或继续播放视频

双眼同时闭上6秒再睁开：关闭视频播放程序界面或网站

单闭左眼（右眼）一下：视频倒退（快进）一下（‘一下’的时长根据视频软件、网站等自有的设定各有不同）

持续单闭左眼（右眼）：视频持续倒退（快进）

0.2.2 任务分工

1.彭佳颖

负责编程任务 1 并完成项目报告第二部分中相应的程序报告撰写，在项目眨眼识别出现问题时进行debug。

2.李培伟

负责编程任务 2 并完成项目报告第二部分中相应的程序报告撰写，作为组长整合小组项目内容。

3.唐芷怡

负责编程任务3（相对于1和2较为简易）并完成项目报告除第二部分以外的内容撰写，制作demo。

一. 问题分析和基本模块

1. 问题分析

我们的三个主要任务说明我们主要需要解决以下三个问题。

1. 面部 68 个特征点的识别如何与视频流的处理结合

“shape_predictor_68_face_landmarks.dat”是基于 dlib 库的开源预学习文件，可以识别出画面中人脸的 68 个标志点。本项目利用相关标志点的坐标建立眨眼相关的判定函数。利用 opencv 库获取摄像头画面，imutils 处理读取出来的帧。对从视屏流中读取出来的帧调整灰度，并用 dlib 的人脸检测识别出相应的关键数据。

2. 选取哪些特征点构造双眼特征值函数，如何由此判定眨眼；如何计算判定闭眼时长

我们想到通过选取眼眶处的特征点，计算上下眼皮的距离和眼睛宽度之比，在程序运行时将此比值和我们提前使用部分程序计算出的正常闭眼（和睁眼）时的值相比较，来判定是否闭眼（眨眼）；通过记录帧数（或判定闭眼的次数）来计算时长。

3. 如何通过程序指令操纵播放软件

我们想到了两种可能的方式——通过某些指令直接控制播放软件或网站，或是通过控制键盘按键来间接操纵。由于前者对我们来说难度过高，而且很难做到在不同软件、网站上适配，因此，尽管后者也有一定的局限性，我们依然选择了通过控制键鼠来进行视频操作。

2. 我们用到的已有模块

2.1. Dlib

Dlib 是一个机器学习的开源库，包含了机器学习的很多算法（比如：图形模型算法； 图像处理：支持读写 Windows BMP 文件，不同类型色彩转换），使用起来很方便，直接包含头文件即可，并且不依赖于其他库（自带图像编解码库源码）。

安装时需要已经安装 Boost 和 Cmake。

2.2. Numpy（Anaconda 自带）

Numpy 是 Python 数据分析的基础，它提供的数据结构比 Python 自身的更高效。Python 自带的列表数据结构，list 列表保存的是对象的指针，比如 [0,1,2] 需要保存 3 个指针和 3 个整数的对象，而 Numpy 是将数据储存在一个连续的内存块中，节约了计算资源。机器学习算法中大部分都是调用 Numpy 库来完成基础数值计算的。

使用 OpenCv 时需要用到。

2.3. OpenCv

OpenCv 是一个图像和视频处理库，用于各种图像和视频分析，如面部识别和检测，车牌阅读，照片编辑，高级机器人视觉，光学字符识别等等。

使用 Numpy 的数组功能，再使用 python-OpenCV。OpenCv 包含 C ++，C，Python 和 Java 的绑定，python-OpenCv 是 Python 特定的 OpenCV 绑定。

使用 OpenCv 主要是为了将视频帧灰度化。

2.4. Imutils

Imutils 是 Adrian Rosebrock 开发的一个 python 工具包，它整合了 OpenCv、numpy 和 matplotlib 的相关操作，主要是用来进行图形图像的处理，如图像的平移、旋转、缩放、骨架提取、显示等等，后期又加入了针对视频的处理，如摄像头、本地文件等。

2.5. Time

Time 模块包含一些可以处理日期和时间或者转换时间格式的内置函数。

使用 time.sleep()函数，可以推迟调用线程的运行，让程序线程暂停休息，传入几秒，休息几秒。

2.6. Argparse

Argparse 是 python 自带的命令行参数解析包，可以用来方便地读取命令行参数。当代码需要频繁地修改参数的时候，使用这个工具可以将参数和代码分离开来，使代码更简洁，适用范围更广。

Argparse 内置于 python，不需要安装。它使我们可以直接在命令行中就向程序中传入参数并让程序运行。

2.7. Scipy.spatial

Scipy 是一个用于数学、科学、工程领域的常用软件包，可以处理插值、积分、优化、图像处理、常微分方程数值解的求解、信号处理等问题。它用于有效计算 Numpy 矩阵，使 Numpy 和 Scipy 协同工作，高效解决问题。

其中，scipy.spatial 的应用领域为空间数据结构和算法。其中最重要的模块应该就是我们用到的距离计算模块 distance 了。矩阵参数每行代表一个观测值，计算结果就是每行之间的 metric 距离。

2.8. Collections

Collections 模块包含了除 list、dict 和 tuple 之外的容器数据类型，如 Counter、defaultdict、Deque、namedtuple、OrderedDict

我们用到的 OrderedDict 是字典子类，它记得其内容被添加的顺序。

2.9. PyAutoGUI

PyAutoGUI 是一个纯 Python 的 GUI（Graphical User Interface 图形用户界面）自动化工具，其目的是可以用程序自动控制鼠标和键盘操作，多平台支持（Windows，OS X，Linux）。

二. 解决方法设计

2.1 导入模块和库

```
from cv2 import cv2
```

```
from imutils.video import FileVideoStream
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import time
import dlib
import imutils
from scipy.spatial import distance as dist
from collections import  OrderedDict
```

2.2 定义函数和字典

2.2.1 定义函数将脸部特征信息转换为数组 array 的格式

```
def shape_to_array(shape, dtype='int'):
    coords = np.zeros((shape.num_parts, 2), dtype=dtype)
    for i in range(shape.num_parts):
        coords[i] = (shape.part(i).x, shape.part(i).y)

    return coords
```

2.2.2 定义计算眼睛长宽比的函数

这里我们利用判断每只眼睛的六个点的坐标，定义了计算眼睛高度和宽度的比值的函数，用以判断眨眼和闭眼动作的发生与否。

```
def accu_angle_eye(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    aspect_ratio = (A + B) / (2.0 *C)

    return aspect_ratio
```

2.2.3 使用 OrderedDict() 构建有序的脸部位置序号字典

```
FACE_LANDMAKE_64_IDEX = OrderedDict([
    ("mouth", (48, 68)),
    ("right_eyebrow", (17, 22)),
    ("left_eyebrow", (22, 27)),
    ("right_eye", (36, 42)),
    ("left_eye", (42, 48)),
    ("nose", (27, 36)),
    ("jaw", (0, 17))
])
```

2.3 设置参数

2.3.1 计数参数与阈值

```
# EAR 的阈值
```

```
BlINK_THRESH = 0.25

# 低于阈值的次数
count_blink1 = 0 #双眼
count_blink2 = 0 #左眼
count_blink3 = 0 #右眼

# 闭眼判定次数
Total = 0
```

此处开始设想为通过获取一帧并判定是否闭眼记为一次，转化成通过计数来判定闭眼时间。

此处曾经出现重大 bug。因为视频和摄像头相应帧数不同导致代码在使用视频做测试文件的同学那里正常运行，但使用摄像头时判定非常不敏感且无报错。【关于 debug 的心路历程：一开始一直怀疑是 videostream 相关代码出现问题，后来将计数代码放在关于帧数的 shreshhold 之前发现可以正常感应眨眼。因此认为是帧数计数相关代码不能“兼容”。】

2.3.2 命令行参数

```
ap = argparse.ArgumentParser()
ap.add_argument('-p', '--shape-
predictor', default='shape_predictor_68_face_landmarks.dat',
                help='the weight to predictor')
ap.add_argument('-v', '--video', type=str, default="camera",
                help="path to input video file")


args = vars(ap.parse_args())
```

因为分工衔接原因，负责判定计数的同学使用的文件素材是 mp4 视频。在此处利用 argparse 的 add_argument 设置衔接时需要频繁改动的参数。（在李培伟同学的运行文件中 default=“camera”默认值设置成相应的 mp4 视频文件）

2.4 检测与获取摄像头

2.4.1 获取检测器

```
# 使用 dlib.get_frontal_face_detector() 获得脸部位置检测器
detector = dlib.get_frontal_face_detector()
# 使用 dlib.shape_predictor 获得脸部特征检测器
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat
")
(lstart, lend) = FACE_LANDMAKE_64_IDEX['left_eye']
(rstart, rend) = FACE_LANDMAKE_64_IDEX['right_eye']
```

2.4.2 检测摄像头并进入判定循环

```
if args['video'] == "camera":
    vs = VideoStream(src=0).start()
    fileStream = False
```

```
else:
    vs = FileVideoStream(args["video"]).start()
    fileStream = True

time.sleep(1.0)

while True:
```

使用 while True 开始循环确保在程序没有用户干预停止的状态下可以一直进行循环。接下来的代码进入每一帧的眨眼判定。

2.5 判定循环

2.5.1 对视频进行循环，读取并处理图片；获取脸部相关信息并画出眼睛轮廓

```
    if fileStream and not vs.more():
        break
# 进行循环，读取图片，并对图片做维度扩大，并进灰度化
    frame = vs.read()
    frame = imutils.resize(frame, width=1200)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #使用 detector(gray, 0)获得脸部位置信息
    rects = detector(gray, 0)
    #循环脸部位置信息，使用 predictor(gray, rect)获得脸部特征位置的信息
    for rect in rects:
        shape = predictor(gray, rect)
        # 将脸部特征信息转换为数组 array 的格式
        shape = shape_to_array(shape)
        # 根据字典，获得左眼和右眼的位置信息
        leftEye = shape[lstart:lend]
        rightEye = shape[rstart:rend]

        # 使用 cv2.convexHull 获得凸包位置，使用 drawContours 画出轮廓位置进行画图操作
        leftContours = cv2.convexHull(leftEye)
        rightContours = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftContours], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightContours], -1, (0, 255, 0), 1)
```

2.5.2 计算得到左右眼，为后面的判断和操作做准备

```
#计算左右眼的 EAR 值
        leftScore = accu_angle_eye(leftEye)
        rightScore = accu_angle_eye(rightEye)
```

2.5.3 进行条件判断，符合闭眼行为和时长后对视频进行相应的操作（关闭、暂停、快进、倒退）

```
#循环，满足条件，增加眨眼次数
```

```
         #关闭操作（双眼闭六秒）
        if leftScore < BlINK_THRESH and rightScore < BlINK_THRESH:
            count_blink1 += 1
        else:
            if count_blink1 >= 30:#计算双眼闭超过六秒
                pyautogui.hotkey('alt','f4') #关闭操作
                count_blink1 = 0
                Total +=1



        #暂停操作（双眼闭两秒）
        if leftScore < BlINK_THRESH and rightScore < BlINK_THRESH:
            count_blink1 += 1
        else:
            if count_blink1 >= 10 and count_blink1 <= 30:#计算双眼闭眼时
间在两秒到六秒之间
                pyautogui.press('space') #暂停操作
                count_blink1 = 0
                Total +=1



        #倒退操作（左眼闭 0.4 秒）
        if leftScore < BlINK_THRESH and rightScore >= 0.20:
            count_blink2 += 1
        else:
            if count_blink2 >= 2:#计算左眼眼闭了 0.4 秒
                pyautogui.press('left') #倒退操作
                count_blink2 = 0
```

2.5.4 进行画图，使用 cv2.putText 展示眨眼次数

```
cv2.putText(frame, 'Blinks: {}'.format(Total), (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
        cv2.putText(frame, 'Left {:.2f}'.format(leftScore), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
        cv2.putText(frame, 'Leftblink {:.2f}'.format(count_blink2), (30
0, 60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
        cv2.putText(frame, 'Right {:.2f}'.format(rightScore), (600, 30)
,
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
        cv2.putText(frame, 'Rightblink {:.2f}'.format(count_blink3), (6
00, 60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 1)
    cv2.imshow('frame', frame)
    if cv2.waitKey(10) & 0xff == 27:
```

```
        break
```

在使用结束后，用 break 结束视频循环。（不然会停不下来）

2.6 结束程序

利用 cv2.destroyAllWindows()关闭所有绘图窗口

```
cv2.destroyAllWindows()
```

三.问题、局限与进一步改善

3.1.问题与局限

1.我们的程序不适用于一边看电脑视频一边低头玩手机（或做其他事）。低头时，眼皮下垂，摄像头捕捉到的眼部上下特征点之间的距离和正对屏幕闭眼时的距离十分相似，我们的程序目前无法区分低头垂眼和闭眼，因此会出现只是拿起手机回了个消息、抬头就发现视频已经被操作了一番的情况。

2.我们程序中判定闭眼的阈值需要根据不同人眼睛的大小在运行前进行调整，也就是说，原程序并不是人人皆可通用的，不同人使用时需要酌情修改程序中“BlINK_THRESH = 0.25”（判定为闭眼的阈值），“rightScore >= 0.20”（单闭左眼时，判定右眼睁眼的阈值），“leftScore >= 0.22”（单闭右眼时，判定左眼睁眼的阈值）处的数值。

3.由于我们是通过键盘操纵视频，因此，当界面上有多个程序时，需要保证视频程序或视频网页是被选定的。

3.2.进一步改善

3.2.1.添加辅助检测特征点

关于问题 1，我认为我们可以通过选取一些面部“辅助检验点”，即不将判定结果全部依赖于眼部数值，添加能够区分出正视（面部正对）屏幕和低头未看屏幕的面部特征值，当同时满足作出闭眼等规定行为和正视屏幕时，才执行命令。

3.2.2.加入机器学习

关于问题 2，要求每个新用户都手动修改阈值是不现实的。首先，有很多用户不懂 python 代码。其次，用户通过观察程序视频框左上角显示的实时特征值来自行判断自己的行为阈值并修改，很容易出现较大误差和错误，而这会导致行为被错误判断、程序灵敏度降低许多。因此，我希望将来可以通过机器学习等方式，在新用户初次运行程序时，进行“自动初始化”，请用户根据程序提示作出相应眨眼、闭眼等行为，程序在这个过程中自动“观察、学习、分析”、记录并使用用户的特征阈值，就像手机面部解锁的初次录入用户面部特征一样。

3.2.3.美化程序界面

目前，我们的程序并没有一个美丽的运行界面，运行之后只会在屏幕上弹出视频流的 frame，朴实无华，如果当时恰好“不修边幅、蓬头垢面”，看到视频框上自己的大脸还会很影响心情。同时，当用户做了动作、指令却没被成功执行时，用户并不能第一时间知道是动作未被识别、程序没能正常运行还是没有选定视频播放程序或网站。后续可做的调整包括隐藏视频框、弹出“动作—指令”提示等等。

致谢

首先，十分感谢 CSDN，Github 等网站以及它们的用户，上传在上面的项目和代码给了我们许多灵感、解决问题的思路和帮助。

其次，十分感谢鲍杨老师给予我们的项目上的指导与这一学期的课程教学。