

换脸项目介绍

项目简介

- 该项目基于世界上最简洁的人脸识别库face_recognition和轻量级 Web 应用框架flask写成face_recognition使用 dlib的最先进的面部识别技术
和深度学习技术构建而成.该模型在 Wild 基准数据集中的Labeled Faces 上的准确性为 99.38%
由于之前不太了解flask框架,所以花费了很多时间了解和学习,最后才比较完美地实现了所需功能

环境配置

- 参见requirements.txt

项目内容

1.项目目标

- 该项目旨在实现一个B/S架构下的换脸功能,总共有三大功能:视频换脸,图片换脸,实时换脸,功能具体情况会在下面阐述

2.基本原理与模型

- face_recognition库可以识别出人脸,并比较两张人脸是否是同一个人,于是我们可以识别视频或图片或摄像头捕捉到的人脸,并将其更换为另一张人脸

3.总体架构设计

采用B/S架构,对每个功能需要写一个或多个前端的html文件,后端需要跟前端进行交互实现每个功能的具体操作并在前端显示

4.项目具体实现

功能实现阐述

- 由于本实验主要着重于python,对前端的html不会过多阐述
- 本实验共有三大功能:视频换脸,图片换脸,实时换脸

视频换脸

- 视频换脸后端主要实现faceswap.py中的函数faceswap_video(video,original_image,swap_image)
- 大致功能如下:
 - 该函数需要把video中出现在original_image中的人脸换成swap_image中的人脸
- 具体函数说明如下:
 - video,original_image,swap_image分别表示原视频路径,需要被换脸图片路径和换入的人脸图片路径,在主函数输入时就已经保证了路径合法性(不可为空的是否为空,文件格式是否正确)
 - original_image可以为空,如果为空,则将video中识别出来的所有人脸替换
 - video和swap_image不可为空,swap_image中如果有多个,以识别出来的第一个人脸为准

- 具体程序说明如下:

- (1)先将swap_image中的人脸提取出来(主函数中已经保证传入的图片中有人脸),如果有多个,以识别出来的第一个人脸为准

```
swap_face=face_recognition.load_image_file(swap_image)
face_locations = face_recognition.face_locations(swap_face)
top_swap, right_swap, bottom_swap, left_swap = face_locations[0]
#将脸提取出来,需要将RGB(face_recognition)改成BGR(opencv)
swap=swap_face[top_swap:bottom_swap, left_swap:right_swap,:-1]
```

- (2)读取视频并获取相关参数,创建一个新视频用于输出最终结果

```
# Open the input movie file
input_movie = cv2.VideoCapture(video)
length = int(input_movie.get(7))
video_width = int(input_movie.get(3))
video_height = int(input_movie.get(4))
video_fps = int(input_movie.get(5))
# Create an output movie file (make sure resolution/frame rate matches input video!)
fourcc = cv2.VideoWriter_fourcc(*'FMP4')
output_movie = cv2.VideoWriter('output.avi', fourcc, video_fps, (video_width, video_height))
```

- (3)获取original_image中所有人脸的编码作为已知脸

```
if original_image == '':
    known_faces = []
else :
    original = face_recognition.load_image_file(original_image)
    original_face_encoding = face_recognition.face_encodings(original)
    known_faces = original_face_encoding
```

- (4)将视频的每一帧提取出来进行操作,对每一帧图片,提取出face_locations和face_encodings

- 对face_encodings中的每一个人脸,如果已知脸不为空,则判断其和某一个已知脸相匹配,匹配则需要替换;如果已知脸为空,则默认需要替换
- 替换时,获取被换的人脸的四个方位,将换入的脸resize成被换的人脸大小再进行替换

- 最后向output_movie中写入该帧

```
face_locations = []
face_encodings = []
frame_number = 0
while True:
    # Grab a single frame of video
    ret, frame = input_movie.read()
    frame_number += 1

    # Quit when the input video file ends
    if not ret:
        break

    # Find all the faces and face encodings in the current frame of video
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)

    for face_num in range(len(face_encodings)):
        # 如果已知脸不为空,判断是否是需要被换的人脸,否则全部替换
        if known_faces:
            match = True in (face_recognition.compare_faces(known_faces, face_encodings[face_num], tolerance=0.50))
        else:
            match= 1
        if(match):
            (top, right, bottom, left) = face_locations[face_num]
            frame_face=cv2.resize(swap,(right-left,bottom-top))
            frame[top:bottom, left:right]=frame_face

    # Write the resulting image to the output video file
    print("Writing frame {} / {}".format(frame_number, length))
    output_movie.write(frame)
```

- (5)由于视频换脸所需时间较长,我们希望让用户能够实时预览进度,所以在处理好一帧后将其resize成合适大小并编码,然后使用yield将每一帧图片编码实时返回

```
frame=resize(frame)
imgencode=cv2.imencode('.jpg',frame)[1]
stringData=imgencode.tostring()
yield (b"--frame\r\n"
      b'Content-Type: text/plain\r\n\r\n'+stringData+b'\r\n')
```

- 在main.py中,我们需要接收返回的图片编码并传递给前端显示

```
@app.route('/faceswap_video_response',methods=['GET','POST'])
def faceswap_video_response():
    try:
        return Response(faceswap_video(session['original_video'],session['original_face'],session['swap_face']),
                       mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
        return Response('static/404.png', mimetype='image/jpeg')
```

- 在templates/faceswap_picture.html中接收并显示:

```

```

- (6)最后释放资源并返回完成图像('static/complete.jpg')通知用户已经完成

```
input_movie.release()
cv2.destroyAllWindows()
complete_image=face_recognition.load_image_file('static/complete.jpg')
complete_image=cv2.resize(complete_image,(800,450))
#转换成opencv中的BGR格式
complete_image=complete_image[:, :,::-1]
imgencode=cv2.imencode('.jpg',complete_image)[1]
stringData=imgencode.tostring()
yield (b"--frame\r\n"
      b'Content-Type: text/plain\r\n\r\n'+stringData+b'\r\n')
```

- 和视频换脸类似,后端主要实现faceswap.py中的函数faceswap_picture(original_image,be_swapped_image,swap_image)
- 大致功能如下:
 - 该函数需要将original_image中出现在be_swapped_image中的人脸换成swap_image中的人脸
- 具体函数说明如下:
 - original_image,be_swapped_image,swap_image分别表示原图片路径,需要被换脸图片路径和换入的人脸图片路径,在主函数输入时就已经保证了路径合法性(不可为空的不为空,文件格式正确)
 - be_swapped_image可以为空,如果为空,则将original_image中识别出来的所有人脸替换
 - original_image和swap_image不可为空,swap_image中如果有多个,以识别出来的第一个人脸为准
 - 具体程序说明如下:
 - (1)和faceswap_video类似,先将swap_image中的人脸提取出来(主函数中已经保证至少有一个人脸).如果有多个,以识别出来的第一个人脸为准

```
swap_face=face_recognition.load_image_file(swap_image)
swap_face_locations = face_recognition.face_locations(swap_face)
top_swap, right_swap, bottom_swap, left_swap = swap_face_locations[0]
swap=swap_face[top_swap:bottom_swap, left_swap:right_swap]
```

- (2)获取be_swapped_image中所有人脸的编码作为已知脸
 - 如果be_swapped_image不为空但没有检测到人脸,则返回2

```
if face_be_swapped_image == '':
    known_faces = []
else :
    be_swapped_face = face_recognition.load_image_file(face_be_swapped_image)
    be_swapped_face_encodings = face_recognition.face_encodings(be_swapped_face)
    known_faces = be_swapped_face_encodings
```

- (3)提取原图片中所有人脸信息(主函数中已经保证原图片中至少有一个人脸)

```
original_face=face_recognition.load_image_file(original_image)
original_face_locations=face_recognition.face_locations(original_face)
original_face_encodings=face_recognition.face_encodings(original_face)
```

- (4)对原图片中每一个人脸,当已知脸不为空时,判断是否和已知脸相匹配,匹配则用于替换,已知脸为空则默认匹配
 - 替换时提取出被换的人脸方位,再将换入的人脸resize成相同大小进行替换

```

for face_num in range(len(original_face_encodings)):
    if(known_faces):
        match = True in (face_recognition.compare_faces
                          (known_faces, original_face_encodings[face_num], tolerance=0.50))
    else:
        match = 1
    if match:
        top, right, bottom, left = original_face_locations[face_num]
        original_face[top:bottom, left:right]=cv2.resize(swap,(right-left,bottom-top))

```

- (5)最后输出换脸后图片,输出后需要给用户看预览图,所以将换脸后图片其resize成合适大小之后编码返回

```

#需要将RGB(face_recognition)改成BGR(opencv),这样才能在最后opencv写入图片时正常显示
original_face = original_face[:, :, ::-1]
cv2.imwrite("output.jpg",original_face)

original_face=cv2.resize(original_face,(800,450))
imgencode=cv2.imencode('.jpg',original_face)[1]
stringData=imgencode.tostring()
return (b"--frame\r\n"
       b'Content-Type: text/plain\r\n\r\n'+stringData+b'\r\n')

```

- 在main.py中接收图片编码并传递给前端显示

```

@app.route('/faceswap_picture_response',methods=['GET','POST'])
def faceswap_picture_response():
    try:
        return Response(faceswap_picture(session['original_face'],session['be_swapped_face'],session['swap_face']),
                      mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
        return Response('static/404.png', mimetype='image/jpeg')

```

- 在templates/faceswap_picture.html中,接收图片并显示

```

```

实时换脸

实时换脸需要从摄像头捕捉人脸, 并将捕捉到的一帧中的人脸替换为服务器预制的卡通图片(本实验中使用京剧的脸谱), 其重要功能由 `capture.py` 中实现.

该功能由类 `CameraSwap` 实现, 该类的对象调用 `one_frame` 函数将返回一个迭代器, 每此迭代将从摄像头捕获图像, 将其中人脸换为预制的脸谱, 返回替换后的图片.

资源管理

实时换脸需要管理摄像头的开启与关闭, 在不使用摄像头时不可独占资源. python 的上下文管理协议在使用 flask 框架时不易使用, 因而没有依靠实现上下文管理协议, 而是采用手动管理.

当 `CameraSwap` 构造时自动获得摄像头资源, 在其不再使用时, 要调用其 `release` 函数释放资源, 此后, `one_frame` 返回的迭代器也将截止.

实时替换

1. 利用 `cv2` 库来进行摄像头的捕获, 其图像存储格式为一个 3 维的 `numpy` 数组(`ndarray`), 前两维长度分别为高度与宽度, 最后一个维度长为 3, 分别代表 BGR(蓝, 绿, 红), 而 `face_recognition` 库使用 RGB 格式, 所以在二者之间需要利用 `numpy` 的索引操作 `[:, :, ::-1]` 将第三个维度逆转.
2. 实时替换需要保证操作的速度, 以免操作时间过长导致帧率过低. 为此, 在对图像进行操作(寻找人脸位置)之前可以将图像缩小, 高度与宽度均变为原本的 $\frac{1}{4}$

```
small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
```

其二, 不对图像进行编码, 直接使用 `face_recognition.face_locations` 来计算所有人脸位置.

3. 换脸时要考虑仅仅替换掉脸部而非将上一步得到的矩形直接替换, 所以在 `CameraSwap` 构造时, 要同时加载脸谱图像的 alpha 通道, 则 `ndarray` 的最后一个维度长度变为 4. 对于人脸所在的矩形位置, 仅仅将脸谱变换至矩形大小后 alpha 通道不为 0 的像素进行替换. 为了不影响处理的速度, 不使用 python 的循环, 而使用 `numpy` 库的三元运算通函数 `np.where(cond, src1, src2)`, 其三个参数规模相同时, 对应位置, 若 `cond` 为 `True`, 则返回值对应位置为 `src1` 的对应位置元素, 反之为 `src2` 对应位置元素. 以下 `swap[:, :, 3]` 即脸谱的 alpha 通道, `i` 取 0, 1, 2(BGR).

```
frame[top:bottom, left:right, i] = np.where(swap[:, :, 3] == 0,
    frame[top:bottom, left:right, i], swap[:, :, i])
```

主函数说明

- 使用flask库进行网页开发,函数说明如下:
 - menu:主菜单,显示初始的菜单界面,总共三个按钮,点击对应的按钮就可以通过重定向到对应的页面

```
@app.route('/', methods=['GET', 'POST'])
def menu():#主菜单
    if request.method == 'POST':
        if 'picture_change_face_btn' in request.form:
            return redirect(url_for('pcf'))
        elif 'video_change_face_btn' in request.form:
            return redirect(url_for('vcf'))
        elif 'camera_change_face_btn' in request.form:
            return redirect(url_for('ccf'))
    return render_template('menu.html')
```

- 主函数视频换脸部分如下:
 - vcf:接收用于视频换脸的文件,需要检测其合法性
 - 需要original_video,original_face,swap_face
 - 对于original_video,需要检查其是否非空,文件名格式是否符合视频格式,在都正确的情况下将其存入一个临时文件夹
 - 对于original_face,当其不为空时检查文件名是否符合图片格式,如果符合且非空就将其存入临时文件夹,然后判断该图片中是否含有脸,没有则提示用户
 - 对于swap_face,检查其是否非空,文件名格式是否符合图片格式,在都正确的情况下将其存入一个临时文件夹,然后判断该图片中是否含有脸,没有则提示用户
 - 最后将临时文件夹中的路径用session保存,用render_template转到templates/faceswap_video.html中
 - 在视频换脸中已经说明,faceswap_video.html中会获取视频换脸中每一帧图片并显示

```
<br>
```

```

@app.route('/faceswap_video_response',methods=['GET','POST'])
def faceswap_video_response():
    try:
        return Response(faceswap_video(session['original_video'],session['original_face'],session['swap_face']),
                      mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
        return Response('static/404.png', mimetype='image/jpeg')

```

- 主函数图片换脸部分如下:

- pcf:接收用于图片换脸的文件,需要检查其合法性
 - 需要original_face,be_swapped_face,swap_face
 - 对于original_face,需要检查其是否非空,文件名格式是否符合图片格式,在都正确的情况下将其存入一个临时文件夹
 - 对于be_swapped_face,当其不为空时检查文件名是否符合图片格式,如果符合且非空就将其存入临时文件夹,然后判断该图片中是否含有人脸,没有则提示用户
 - 对于swap_face,检查其是否非空,文件名格式是否符合图片格式,在都正确的情况下将其存入一个临时文件夹,然后判断该图片中是否含有人脸,没有则提示用户
 - 最后将临时文件夹中的路径用session保存,用render_template转到templates/faceswap_picture.html中
- 在图片换脸中已经说明,faceswap_picture.html会获取最后的换脸图片并显示

```
<br>
```

```

@app.route('/faceswap_picture_response',methods=['GET','POST'])
def faceswap_picture_response():
    try:
        return Response(faceswap_picture(session['original_face'],session['be_swapped_face'],session['swap_face']),
                      mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
        return Response('static/404.png', mimetype='image/jpeg')

```

- 主函数实时换脸部分如下:

- ccf:

- 按下enter按钮后会创建一个CameraSwap用于实时换脸,用render_template转到templates/camera_change_face.html;
- 按下back按钮会释放camera并退回到主页面

```
@app.route('/ccf', methods=['GET', 'POST'])
def ccf():
    global camera
    if request.method=='POST':
        if 'enter_btn' in request.form:
            if camera == None:
                camera = CameraSwap()
            return render_template('camera_change_face.html')
        if 'back_btn' in request.form:
            if camera != None:
                camera.release()
                camera = None
            return redirect(url_for('menu'))
    return render_template("ccf.html")
```

- 在templates/camera_change_face.html中获取摄像头捕获并换脸后的图片之后显示

```

```

```
@app.route('/camera_change_face', methods=['GET', 'POST'])
def camera_change_face():
    try:
        return Response(camera.one_frame(), mimetype='multipart/x-mixed-replace; boundary=frame')
    except Exception as e:
        return Response('static/404.png', mimetype='image/jpeg')
```

5.运行结果

1.在终端运行

```
python main.py
```

2.在浏览器打开127.0.0.1:5000



3.图片换脸

- (1)进入图片换脸

图片换脸

图片换脸会将原图片中特定的人脸(被换的人脸)换成您指定的人脸(换入的人脸),上传文件具体需求如下:

请注意:1.原图片和换入的人脸图片不可为空

2.被换的人脸图片可以为空(即可以不上传),如果为空,则将视频中检测出的所有人脸替换

3.被换的人脸图片中检测出来的所有人脸都将被换

4.换入的人脸图片如果有多人,则将检测出的第一个人作为换入的人脸

选择原始图片

选择被换的人脸图片

选择换入的人脸图片

开始换脸

返回

- (2)上传对应文件并点击开始换脸

- 原图片:two_people.jpg
- 被换的人脸图片:biden.jpg
- 换入的人脸图片:alex-lacamoire.png

图片换脸

图片换脸会将原图片中特定的人脸(被换的人脸)换成您指定的人脸(换入的人脸),上传文件具体需求如下:

请注意:1.原图片和换入的人脸图片不可为空

2.被换的人脸图片可以为空(即可以不上传),如果为空,则将视频中检测出的所有人脸替换

3.被换的人脸图片中检测出来的所有人脸都将被换

4.换入的人脸图片如果有多人,则将检测出的第一个人作为换入的人脸

C:\fakepath\two_people.jpg

C:\fakepath\biden.jpg

C:\fakepath\alex-lacamoire.png

开始换脸

返回

- (3)显示换脸结果(结果保存在py文件同目录下'output.jpg'):

图片换脸ing

换脸结果将保存至py文件同目录下'output.jpg'

点击页面最下方的返回键可以返回图片换脸界面

换脸预览(预览图片大小为800*450)马上为您呈现
请等待...



返回

4.视频换脸

- (1)进入视频换脸

视频换脸

视频换脸会将原视频中特定的人脸(被换的人脸)换成您指定的人脸(换入的人脸),上传具体需求如下:

- 1.原视频和换入的人脸图片不可为空
- 2.被换的人脸图片可以为空(即可以不上传),如果为空,则将视频中检测出的所有人脸替换
- 3.被换的人脸图片中检测出来的所有人脸都将被换
- 4.换入的人脸图片如果有多人,则将检测出的第一个人作为换入的人脸

选择原始视频

选择被换的人脸图片

选择换入的人脸图片

开始换脸

返回

- (2)上传对应文件并点击开始换脸

- 原视频:very_short_hamilton_clip.mp4
- 被换的人脸图片:空
- 换入的人脸图片:swap.jpeg

视频换脸

视频换脸会将原视频中特定的人脸(被换的人脸)换成您指定的人脸(换入的人脸),上传具体需求如下:

- 1.原视频和换入的人脸图片不可为空
- 2.被换的人脸图片可以为空(即可以不上传),如果为空,则将视频中检测出的所有人脸替换
- 3.被换的人脸图片中检测出来的所有人脸都将被换
- 4.换入的人脸图片如果有多人,则将检测出的第一个人作为换入的人脸

C:\fakepath\very_short_hamilton_clip.mp4

选择被换的人脸图片

C:\fakepath\swap.jpeg

开始换脸

返回

- (3)实时显示换脸结果,截图如下:

视频换脸ing

换脸结果将保存至py文件同目录下'output.avi'

点击页面最下方的返回键可以返回视频换脸界面

换脸预览(预览图片大小为800*450)马上为您呈现
视频换脸可能等待时间较长,请耐心等待...



返回

- (4)换脸完成后显示完成图片(结果保存在py文件同目录下'output.avi'):

视频换脸ing

换脸结果将保存至py文件同目录下'output.avi'

点击页面最下方的返回键可以返回视频换脸界面

换脸预览(预览图片大小为800*450)马上为您呈现

视频换脸可能等待时间较长,请耐心等待...



返回

5.实时换脸

- (1)进入实时换脸

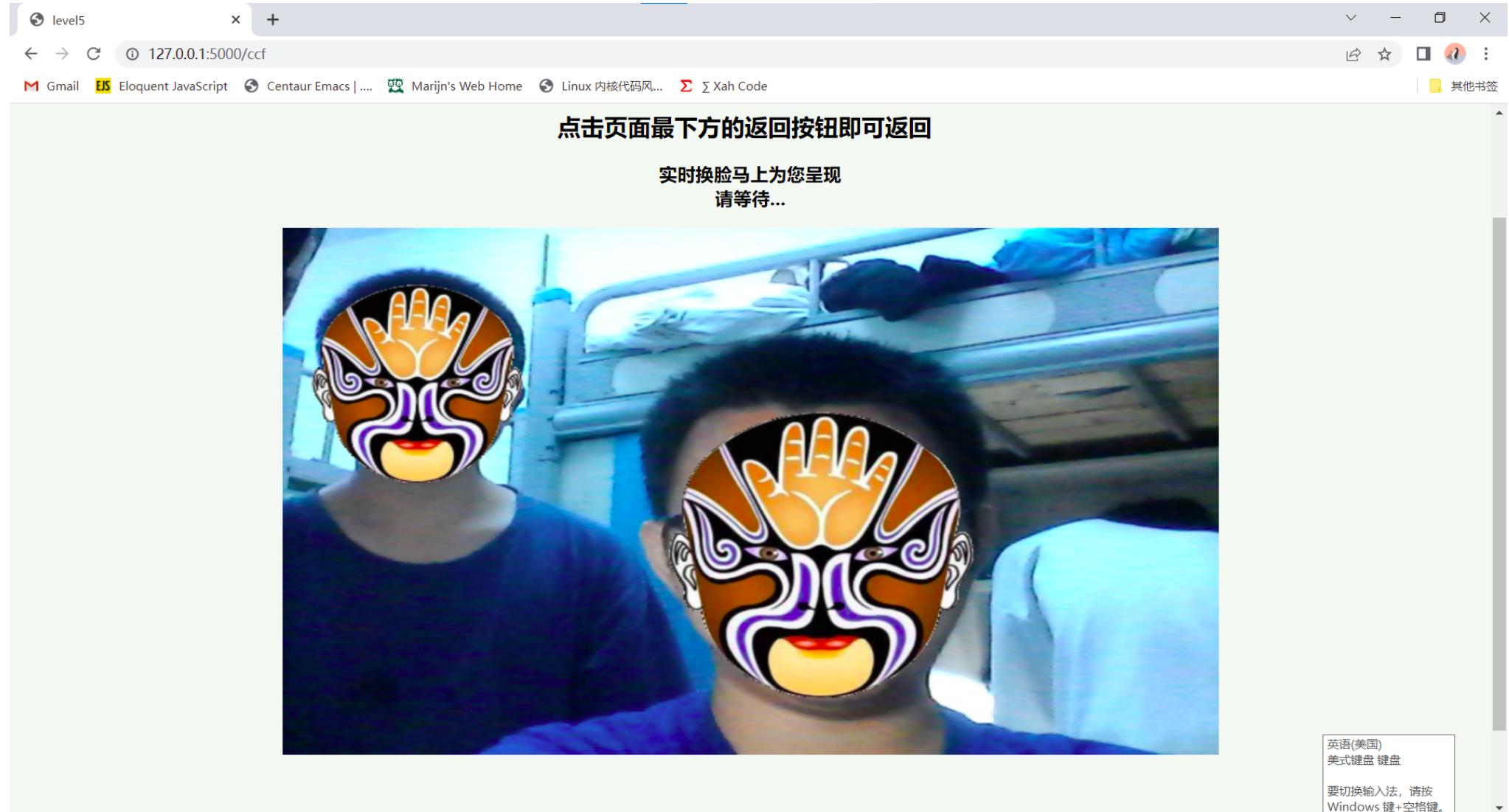
实时换脸

实时换脸将打开您的摄像头实时为您换脸
点击开始换脸按钮即可开始

开始换脸

返回

- (2)点击开始换脸按钮后开始实时换脸,截图如下:



项目分析-总结与展望

本次实验中我们学习了flask架构,学习了html和少许的js,让我们第一次体会到了网页开发的感觉,最后加上模板的渲染,我个人感觉效果还不错,虽然功能没有那么的丰富,但是基础的开发过程我们都已经有所了解,希望以后有机会能够开发更大更好、让许多人都可以喜欢的项目吧