《多媒体技术》实验报告

2220193626 网络二班李子毅

一、 实验选题

图像处理应用系统的开发

二、 实验要求

设计一个应用系统实现相关功能，提供原始文件库，以及测试库。对系统的功能可以进行定量的评价，并可以逐步演示功能模块。

三、 实验内容

1. 平台选择

对于本实验，我选择的编程语言是 python，开发平台是 pycharm+anaconda，选择 python 作为我的开发语言的原因是，python 是一种语法简单的脚本化语言，python 可以调用很多第三方库，包括我采用的当下比较热门的图像处理库 OpenCV，学习成本较低，代码可读性也很高。选择 pycharm 的原因是，比起 python 自带的 IDE，它内置了语法补全、语法高亮的功能的功能，最新版的 pycharm 也可以一键安装所需的 python 版本，非常方便，同时加上 anaconda，它包含了许多开发需要的 python 科学包，如 numpy 等，并且可以方便的使用 conda 命令，比起 pip 命令更加方便。

2. **处理技术介绍**（如果是算法设计，需要给出算法的基本思想介绍；如果是媒体处理，需要给出处理的基本思路）

（1）阈值二值化：这里我采用的是二分阈值化，实现非常简单，首先，图像必须转化为灰度图像，对于 RGB 空间而言，需要将他们取平均，得到的为灰度值，对这个灰度图像设置一个阈值，超过这个阈值的像素点，显示为黑色图像，低于这个阈值的像素点设置为白色，阈值范围在[0,255]。

（2）边缘检测：边缘是指图像中图像像素值变化明显的位置。这些变化明显的位置常常是图像中需要关注的位置，对这些关键位置的检测，就叫做边缘检测。也就是需要对这些位置求导数，这里一般有两种常用的算子，我采用了 canny 算子，即梯度方向的二阶导数过零点，但首先需要使用高斯滤波来降噪，以排除噪点的影响（为了说明高斯滤波降噪的重要性，我设置了一个 check 方法来对比），接着使用 canny 算子来计算边缘部分， 在获得梯度大小和方向后，将对图像进行全面扫描，以去除可能不构成边缘的所有不需要的像素。为此，在每个像素处，检查像素是否是其在梯度方向上附近的局部最大值，最后采用双阈值法得到最终图像，它的原理是，在非极大值抑制后的边缘点中，设置两个阈值 TH 和 TL，梯度值超过 TH 的为强边缘，小于 TH 大于 TL 的为弱边缘，小于 TL 的不是边缘，可以肯定的是，强边缘必然是边缘点，因此必须将 T1 设置的足够高，以要求像素点的梯度值足够大（变化足够剧烈），而弱边缘可能是边缘，也可能是噪声，如何判断呢？当弱边缘的周围 8 邻域有强边缘点存在时，

就将该弱边缘点变成强边缘点，以此来实现对强边缘的补充。最终得到了一张边缘检测之后的视频图像。

（3）轮廓检测：算法是将所有沿着相同的连续点的曲线设置为相同的颜色，首先把图像去噪处理，然后转化成二值图，对于这个二值化的图像，掏空内部点，绘制轮廓，然后将绘制出的轮廓在原图像上显示即可。

（4）图像滤波：在尽量保持图像原始特征的前提下，对图像的噪声进行抑制，原理上是采用一个滤波器进行滤波操作，滤波是一个邻域算子，根据周围像素点的值来确定该点像素点的值，这里我实现了两种滤波操作，高斯滤波和均值滤波，均值滤波的实现方式比较简单，就是对一个像素去 5*5 的像素和，再取平均数，用这个平均数去替代这个像素点的值，直至所有像素点均完成迭代，高斯滤波是采用一种线性平滑滤波器，对于服从正态分布的噪声有很好的抑制作用。对比均值滤波，其不同在于它的滤波器的模板系数随着距离模板中心的增大而减小（服从二维高斯分布）

（5）色彩转换，这个功能较为简单，主要是实现了不同色彩空间的转化功能，我选择实现的是 RGB 到 HSV 的颜色空间转化功能，然后通过调控色调和饱和度，再映射回 RGB 空间可以看到色彩转换后的结果。

（6）提高对比度：首先通过计算得到一个对比度阈值，然后将其进行（8,8）的直方图均衡化操作，然后将这个操作应用到每一帧的图像上，采用的同样是先转化为其他的颜色模型，这里我选择转化为 lab 模型，然后进行对比度增强操作，再转化到 RGB 空间即可

（7）人脸识别和笑脸识别：人脸识别主要是采用基于 harr 特征的级联分类器进行实现，级联分类器的函数是通过大量带人脸和不带人脸的图片通过机器学习得到的。对于人脸识别来说，需要几万个特征，通过机器学习找出人脸分类效果最好、错误率最小的特征。训练开始时，所有训练集中的图片具有相同的权重，对于被分类错误的图片，提升权重，重新计算出新的错误率和新的权重。直到错误率或迭代次数达到要求。这种方法叫做 Adaboost，最终的分类器是一个强分类器，它是众多弱分类器的组合，实现上首先调用分类器函数选出人脸区域，然后再把人脸区域绘制出来即可，同理如果要进行人眼检测，就要用训练好的人眼的配置文件，然后基础区域选择人脸区域，画出人眼即可。

（8）更加精确的人脸识别：在做人脸识别的过程中，由于 OpenCV 内置的级联分类器只是靠机器学习原理进行分类，虽然即便是可以经过自己的样本经过正样本和负样本的训练，但经训练后发现效果仍旧不好，因此，我决定采用深度学习的 keras 和 TensorFlow 框架搭建的 CNN(卷积神经网络)去训练一个自己的分类器。训练过程采用我自己的 1000 张样本以及同学的 1000 张样本实现，训练时采用交叉验证的思想随机划分数据集，因为是小样本学习，训练过程采用我的 cpu，最终训练的时长为 2h 左右，使用时首先采用 OpenCV 框选出人脸区域，然后将人脸区域传入得到对应的分类标签，再根据标签来判断是自己还是他人。

3. 本实验的设计性和创新性体现（如果是算法设计，说明算法的改进和创新之处及其依据；如果是处理技术，说明新颖的处理效果，及其对应的方法介绍）（正文部分）

(1) 将 OpenCV 这一 python 大名鼎鼎的图像处理库与当下比较简易的图形化界面技术相结合，做到了可以让用户通过简单的操作来直观地感受课堂中讲授的部分图像处理技术的结果，如高斯滤波，阈值二分类、边缘检测等，操作简单且不同功能之间的切换迅速，代码的编写也经过多次重构，纯按照相关原理编写，有利

于后续的学习。

(2) 考虑到 OpenCV 自带的级联分类器的人脸识别作用的局限性，即只能识别出有人脸，但不能识别出是谁的人脸，且识别的准确度也有待提升，因此了解了 CNN 在人脸识别中的用法，当把图像矩阵化后，规定一个权值矩阵，这个权值与图像结合保证每个像素点都被结合一起，并经过卷积运算得到一个新的矩阵，经过多次卷积运算后的图像进入池化层，对于卷积后的图像只按照规则取部分像素点，该操作用于减少训练量，然后全连接层用于生成和输入时相同量的像素图，最后输出层加入损失函数的计算，用于预测误差，一旦前向传播完成，反向传播就会开始更新权重与偏差，以减少误差和损失。而后自己采集数据并编写程序训练了一个 CNN 的人脸识别分类器，可以识别出某一张图像是不是本人，可以与 OpenCV 的识别情况进行对比，直观地感受区别。

4. 实现步骤（如果是代码实现，给出关键功能模块的代码；如果是处理技术，给出处理步骤）

（正文部分）

(1)图像处理系统主题部分代码

```
#导入相关包
import PySimpleGUI as sg
import numpy as np
import cv2
from predict import *
# 打开内置摄像头
cap = cv2.VideoCapture(0)
# 背景色
sg.theme('Reddit')
# 定义窗口布局，创建不同功能的按钮
layout = [
    [sg.Image(filename='', key='image')],
    [sg.Radio('默认状态', 'Radio', True, size=(10, 1))],
    [sg.Radio('阈值二值化', 'Radio', size=(10, 1), key='thresh'),
     sg.Slider((0, 255), 128, 1, orientation='h', size=(40, 15), key='thresh_slider')],
    [sg.Radio('边缘检测', 'Radio', size=(10, 1), key='canny'),
     sg.Slider((0, 255), 128, 1, orientation='h', size=(20, 15), key='canny_slider_a'),
     sg.Slider((0, 255), 128, 1, orientation='h', size=(20, 15), key='canny_slider_b'),
     sg.Checkbox('加入降噪',key='gass'),
     ],
    [sg.Radio('轮廓检测', 'Radio', size=(10, 1), key='contour'),
     sg.Slider((0, 255), 127, 1, orientation='h', size=(40, 15), key='base_slider')],
    [sg.Radio('高斯滤波', 'Radio', size=(10, 1), key='blur'),
     sg.Slider((1, 11), 1, 1, orientation='h', size=(40, 15), key='blur_slider'),
     sg.Checkbox('线性滤波', key='line'),
     ],
    [sg.Radio('色彩转换', 'Radio', size=(10, 1), key='hue'),
     sg.Slider((0, 225), 0, 1, orientation='h', size=(40, 15), key='hue_slider')],
    [sg.Radio('调节对比度', 'Radio', size=(10, 1), key='enhance'),
```

```
     sg.Slider((1, 255), 128, 1, orientation='h', size=(40, 15),
key='enhance_slider')],
    [sg.Radio('人脸识别','Radio',size=(10,1),key='face'), sg.Radio('笑脸识别
','Radio',size=(10,1),key='smile')],
    [sg.Radio('强化识别','Radio',size=(10,1),key='Big')],
    [sg.Button('Exit', size=(10, 1))]
]
# 窗口设计
window = sg.Window('实时图像处理系统',
                   layout,
                   location=(500, 500),
                   finalize=True,
                   font='宋体',
                   )
# 导入识别模型
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_smile.xml')
##主体部分
while True:
    event, values = window.read(timeout=0, timeout_key='timeout')
    # 实时读取图像,逐帧捕获
    ret, frame = cap.read()
    ############阈值二分类##############
    if values['thresh']:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        #该函数有两个输出，只取第二个，为阈值之后的图像
        frame = cv2.threshold(frame, values['thresh_slider'], 255,
cv2.THRESH_BINARY)[1]
    ############边缘检测##############
    if values['canny']:
        if values['gass']:
            # 加入降噪
            frame = cv2.GaussianBlur(frame, (5, 5), 0)
        #不加入降噪
        frame = cv2.Canny(frame, values['canny_slider_a'], values['canny_slider_b'])
    ############轮廓检测##############
    if values['contour']:
        #需要先转化为二值图
        gray= cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray=cv2.threshold(gray,values['base_slider'],255,cv2.THRESH_BINARY)[1]
        #21 维的高斯滤波降噪
        hue = cv2.GaussianBlur(gray, (21, 21), 1)
```

```
        #三个参数，输入图像、层次类型、轮廓逼近方法，第一个返回值为修改后的图像，也就是轮廓
        cnts = cv2.findContours(hue, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
        #绘制轮廓,cnt 为存储了多个轮廓的数组，color 代表线条的颜色，
        cv2.drawContours(frame, cnts, -1, (0, 0, 255), 3)
    ###########轮廓检测#############
    if values['blur']:
        #高斯滤波
        frame = cv2.GaussianBlur(frame, (21, 21), values['blur_slider'])
        if values['line']:
            kernel=np.ones((5,5),np.float32)/25
            frame=cv2.filter2D(frame,-1,kernel)
    ###########颜色转换#############
    if values['hue']:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        frame[:, :, 0] += int(values['hue_slider'])
        #再映射回 RGB 空间
        frame = cv2.cvtColor(frame, cv2.COLOR_HSV2BGR)
    ###########调节对比度#############
    if values['enhance']:
        ##得到一个提高度
        enh_val = values['enhance_slider'] / 40
        #直方图均衡化的函数进行实例化，选择（8,8）的网格
        clahe = cv2.createCLAHE(clipLimit=enh_val, tileGridSize=(8, 8))
        lab = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
        #将这个函数应用到当前图像上，得到最终图像
        lab[:, :, 0] = clahe.apply(lab[:, :, 0])
        frame = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    ###########人脸识别##################
    if values['face']:
        #导入分类函数，找到人脸的大致位置
        faces = face_cascade.detectMultiScale(frame, 1.3, 2)
        for (x, y, w, h) in faces:
            # 画出人脸框，蓝色，画笔宽度微
            face = cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            # 框选出人脸区域，在人脸区域而不是全图中进行人眼检测，节省计算资源
            face_area = face[y:y + h, x:x + w]
            ## 人眼检测
            # 用人眼级联分类器引擎在人脸区域进行人眼识别，返回的 eyes 为眼睛坐标列表
            eyes = eye_cascade.detectMultiScale(face_area, 1.3, 10)
            for (ex, ey, ew, eh) in eyes:
                # 画出人眼框，绿色，画笔宽度为 1
                cv2.rectangle(face_area, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 1)
    ###########笑脸识别############
```

```
    if values['smile']:
        faces = face_cascade.detectMultiScale(frame, 1.3, 2)
        for (x, y, w, h) in faces:
            face = cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            face_area = face[y:y + h, x:x + w]
            eyes = eye_cascade.detectMultiScale(face_area, 1.3, 10)
            for (ex, ey, ew, eh) in eyes:
                cv2.rectangle(face_area, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 1)
                ## 微笑检测
                # 用微笑级联分类器引擎在人脸区域进行人眼识别，返回的 eyes 为眼睛坐标
列表
                smiles = smile_cascade.detectMultiScale(face_area, scaleFactor=1.16,
minNeighbors=65, minSize=(25, 25),

flags=cv2.CASCADE_SCALE_IMAGE)
                for (ex, ey, ew, eh) in smiles:
                    # 画出微笑框，红色（BGR 色彩体系），画笔宽度为 1
                    cv2.rectangle(face_area, (ex, ey), (ex + ew, ey + eh), (0, 0, 255),
1)
                    cv2.putText(frame, 'Smile', (x, y - 7), 3, 1.2, (0, 0, 255), 2,
cv2.LINE_AA)
    ############强化版人脸识别#################
    if values['Big']:
        #初始化模型
        my_model = keras.models.load_model('D:\model\model.h5')
        #转化为灰度图像
        gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        # 利用分类器识别出哪个区域为人脸
        faces = face_cascade.detectMultiScale(gray, 1.1, 5)
        if len(faces) > 0:
            for (x, y, w, h) in faces:
                # 截取脸部图像提交给模型识别这是谁
                image = frame[y: y + h, x: x + w]
                faceID = face_predict(image, my_model)

                # 如果是“我”
                if faceID == 0:
                    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), thickness=2)
                    # 文字提示是谁
                    cv2.putText(frame, 'me',
                                (x + 30, y + 30),  # 坐标
                                cv2.FONT_HERSHEY_SIMPLEX,  # 字体
                                1,  # 字号
                                (255, 0, 255),  # 颜色
```

```
                                2)  # 字的线宽
                else:
                    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), thickness=2)
                    # 文字提示是谁
                    cv2.putText(frame, 'others',
                                (x + 30, y + 30),  # 坐标
                                cv2.FONT_HERSHEY_SIMPLEX,  # 字体
                                1,  # 字号
                                (255, 0, 255),  # 颜色
                                2)
    """
    暂时停止使用的功能，因为太卡了
    if values['finger']:
        skinCrCbHist = np.zeros((256, 256), dtype=np.uint8)
        cv2.ellipse(skinCrCbHist, (113, 155), (23, 25), 43, 0, 360, (255, 255, 255),
-1)  # 绘制椭圆弧线
        YCrCb = cv2.cvtColor(frame, cv2.COLOR_BGR2YCR_CB)  # 转换至 YCrCb 空间
        (y, Cr, Cb) = cv2.split(YCrCb)  # 拆分出 Y,Cr,Cb 值
        skin = np.zeros(Cr.shape, dtype=np.uint8)  # 掩膜
        (x, y) = Cr.shape
        for i in range(0, x):
            for j in range(0, y):
                if skinCrCbHist[Cr[i][j], Cb[i][j]] > 0:  # 若不在椭圆区间中
                    skin[i][j] = 255
        frame= cv2.bitwise_and(frame, frame, mask=skin)"""
    if event == 'Exit' or event is None:
        break
    # GUI 实时更新，捕捉当前图像，然后转化为 byte 形式
    imgbytes = cv2.imencode('.png', frame)[1].tobytes()
    # 更新图像
    window['image'].update(data=imgbytes)
#关闭 GUI 界面
window.close()
```

(2)CNN 人脸识别部分函数代码：

```
##########预测函数#########################
def face_predict(image,model):
    image = resize_image(image)
    image = image.reshape((1, IMAGE_SIZE, IMAGE_SIZE, 3))
    image=image.astype('float32')
    image/=255
    # 给出输入属于各个类别的概率
    result = model.predict_proba(image)
```

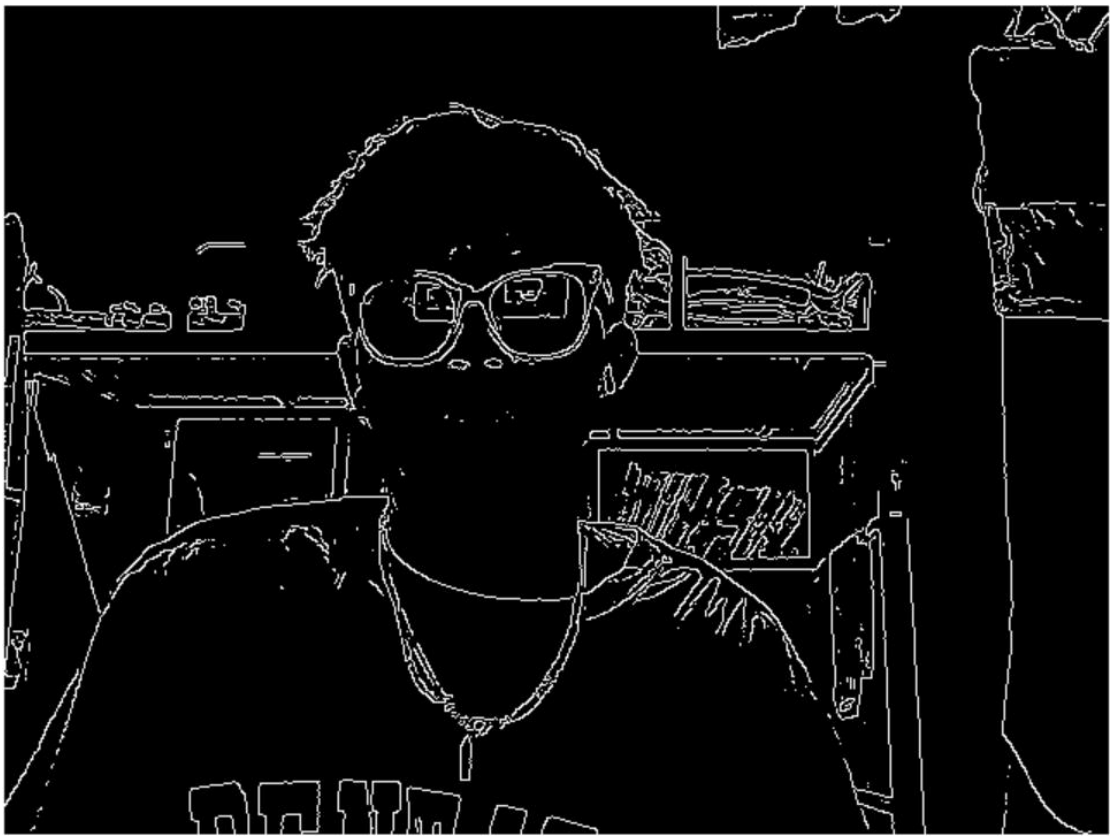
```
    print('result:', result)

    # 给出类别预测：0 或者 1
    result = model.predict_classes(image)

    # 返回类别预测结果
    return result[0]
```

四、 实验效果

（给出应用系统的主要运行过程和结果图，及其简要说明）

(1) 阈值二值化

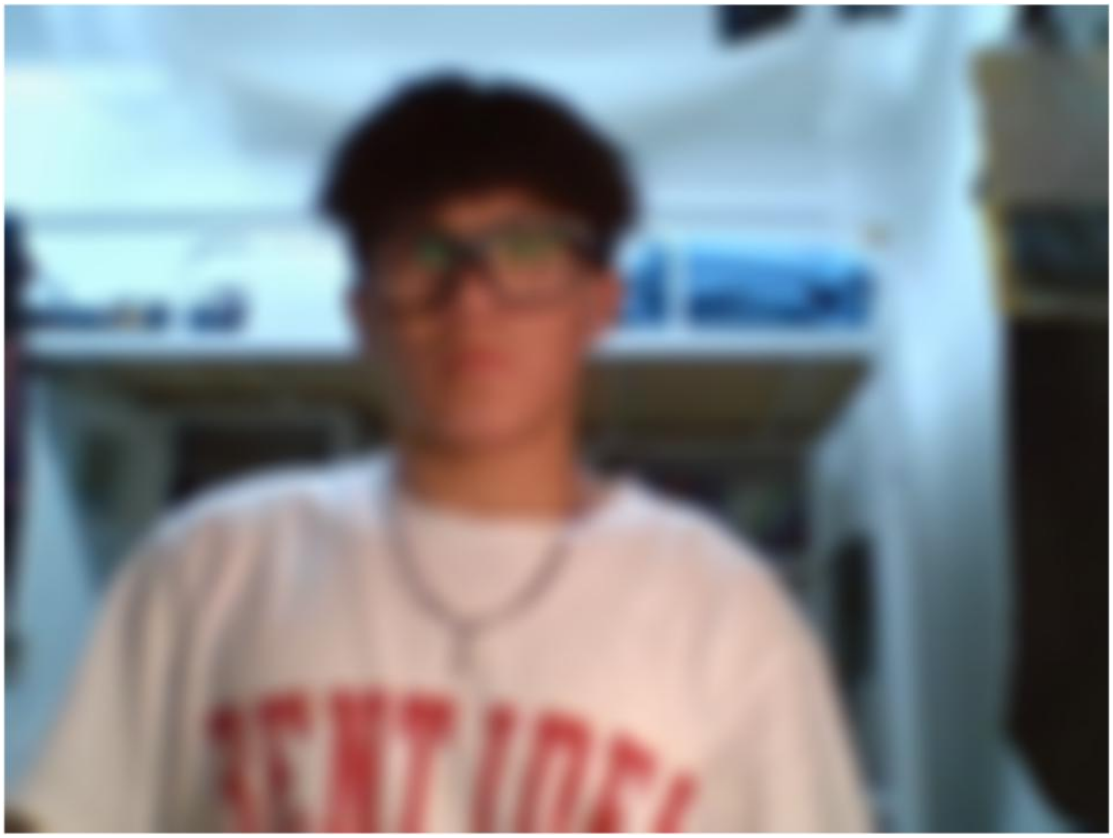
(2) 边缘检测



加入降噪之后：

(3) 轮廓检测

(4) 高斯滤波

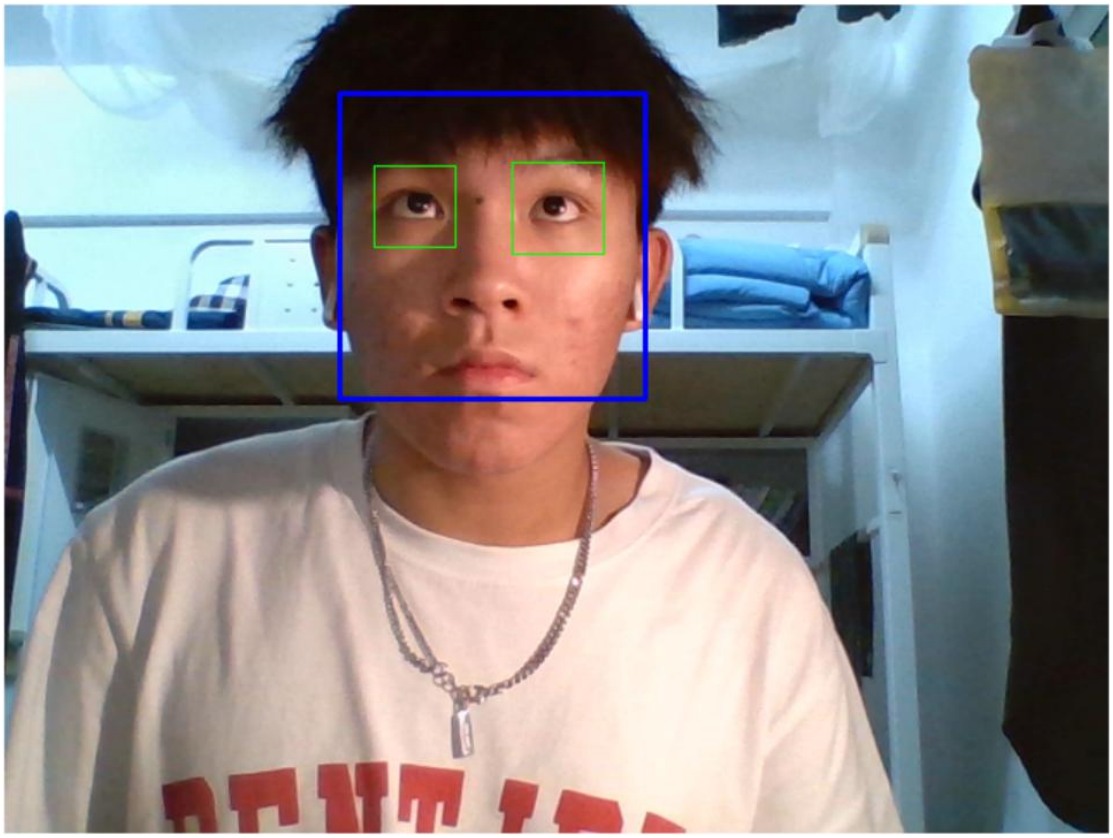


线性滤波：

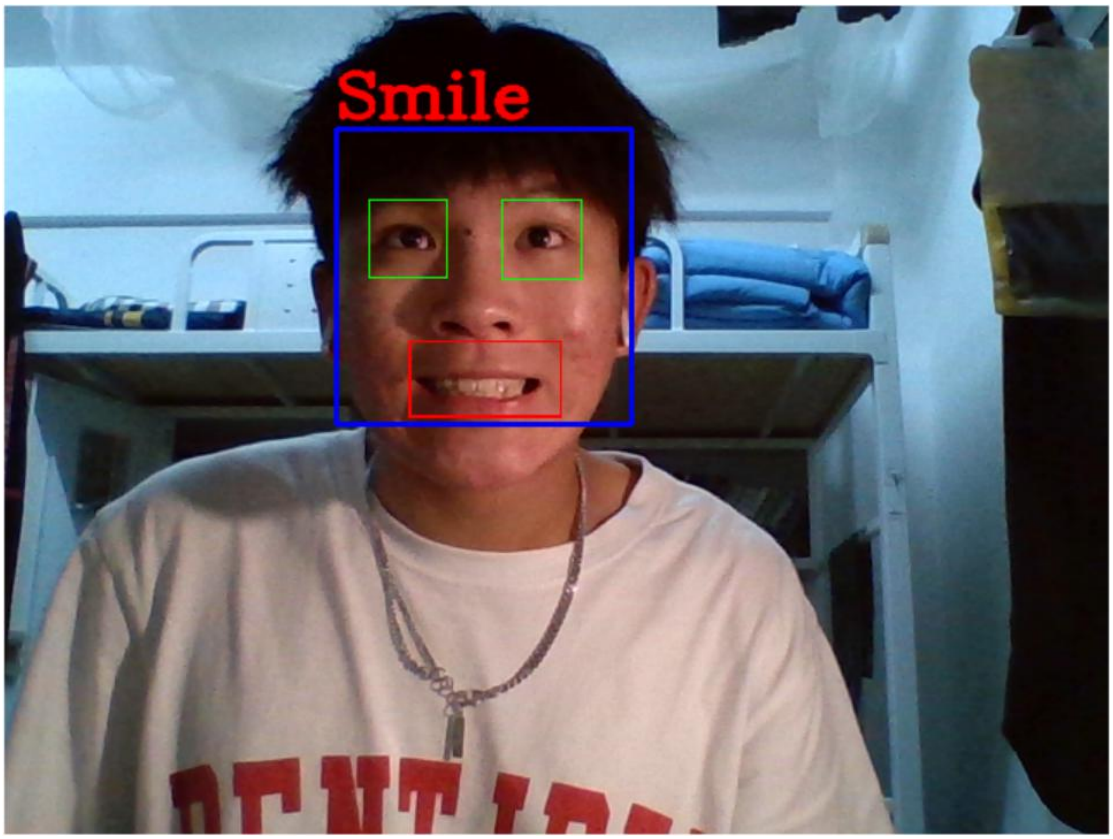
(5) 色彩空间转换:

(6) 调节对比度:

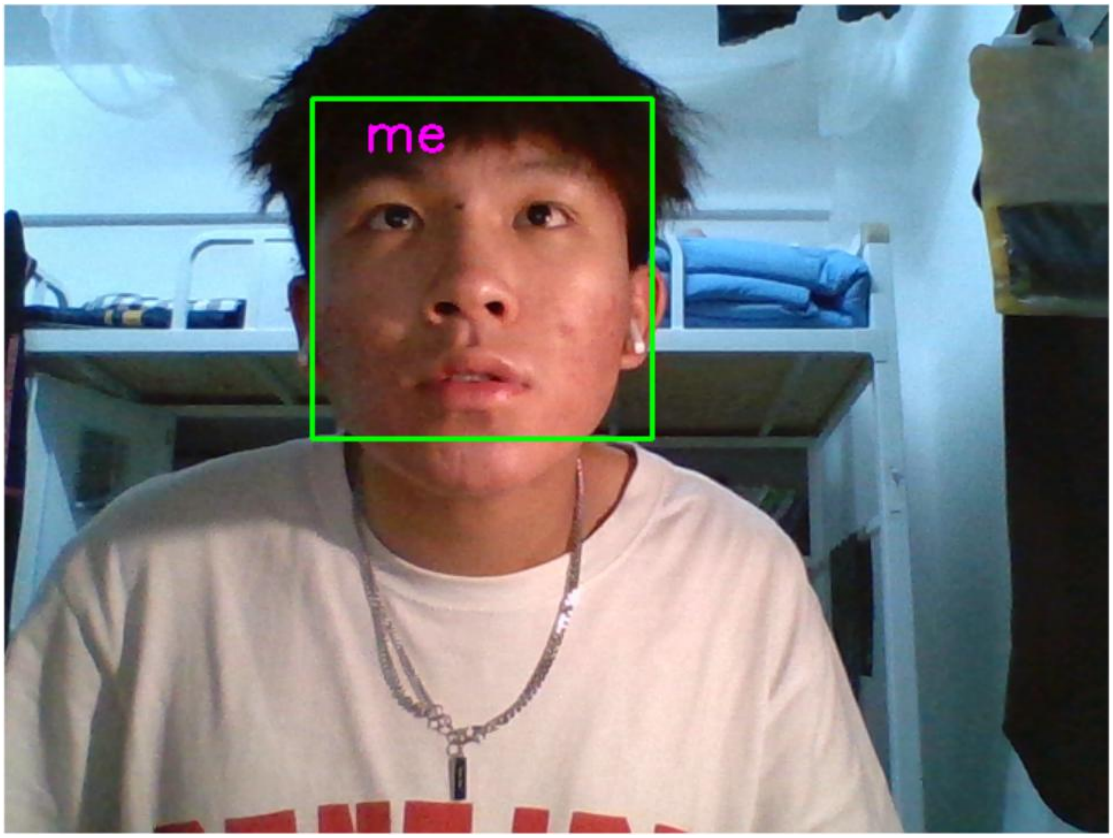
(7) 人脸识别+人眼识别



(8)笑脸识别：



（9）CNN 人脸识别：



五、 实验结论

1. 实验不足与未来展望：
（1）由于为了提升识别精度和效果，事先采用了级联分类器框选了区域，再采用训练好的模型的识别功能，导致该模型可以实现多人脸识别的优点没有显现出来，后续可以考虑把区域划分成整个图像区域，但需要采用更多的样本进行训练
（2）可以将二分类问题推广至多分类问题，实现多人脸分类的操作，理论上只需增加样本，并对不同的样本标签不同即可。另外增加样本可以考虑使用 transformer 模型
（3）可以换用更美观的图形化界面，如 pyqt5、tkinter 等
2. 实验总结：本次实验中我选择了实时图像处理系统的构建这个课题，然后运用 python 代码编写了自己的图像处理系统，实现了上课老师提过的各个功能，有些上课记的不牢固的知识，去查阅了 OpenCV 的官方文档以及相关的博客和论文(如轮廓检测)，同时加深了对视频图像处理的过程的理解。在完成深度学习的人脸识别的过程中，我查阅了 Github 的相关代码，自主配置了环境，花了很长时间，模型的选择也是一个漫长的过程，自主搜索决定了使用 RNN 模型，然后简单学习了 TensorFlow 和 keras 框架的使用方法，编写了人脸识别的代码，效果较 OpenCV 的级联分类器确实有很大提升，这一过程大约准备了两周，其中又翻阅书籍去看自己以前看过的机器学习、深度学习的相关书籍，并将其中的交叉验证等思想运用在模型的训练和检验之中。