

《多媒体技术》实验报告 4

黄勛 22920212204392

1. 运行程序截图和简要说明

1) opencv 的视频捕获与人脸检测功能，代码见 task1.py

运行：



2) 使用 dlib 检测 68 个人脸特征点，代码见 task2.py

运行：



2.主要代码展示和分析

1) opencv 的视频捕获与人脸检测功能，代码见 task1.py

①先安装 cv2

```
问题 1 输出 终端 调试控制台

WARNING: You are using pip version 20.2; however, version 23.1 is available.
PS E:\大二下\多媒体技术\lab4> pip install opencv-python
Using cached opencv_python-4.7.0.72-cp37-abi3-win_amd64.whl (38.2 MB)
Using cached numpy-1.24.2-cp39-cp39-win_amd64.whl (14.9 MB)
Installing collected packages: numpy, opencv-python
Attempting uninstall: numpy
Found existing installation: numpy 1.17.2
Uninstalling numpy-1.17.2:
Successfully uninstalled numpy-1.17.2
Successfully installed numpy-1.24.2 opencv-python-4.7.0.72
```

②找到对应的“haarcascade_frontalface_default.xml”文件

D: haarcascade_frontalface_default - Everything				
文件(F) 编辑(E) 视图(V) 搜索(S) 书签(B) 工具(T) 帮助(H)				
"D:* haarcascade_frontalface_default				
名称	路径	大小	修改时间	
haarcascade_frontalface_default.xml	D:\Program Files\Lib\site-packages\cv2\data	909 KB	2023/4/19 0:47	

③编写代码

```
task1.py X task2.py
task1.py > ...
1 import cv2
2
3 cap = cv2.VideoCapture(0, cv2.CAP_DSHOW) # 首参数camera_id为设备摄像头的id, 一般为0, 如果有多个摄像头,
4
5 classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml") #加载人脸识别分类器, 这里使用
6
7 color = (0, 225, 0) #人脸框的颜色
8
9 while cap.isOpened(): # 循环读取每一帧, 直到读取失败, 即视频播放完毕
10     ret, frame = cap.read() # 读取一帧数据, frame表示摄像头读取的图像矩阵mat类型
11
12     # 人脸检测
13     gray = cv2.cvtColor(frame,
14                             cv2.COLOR_BGR2GRAY) # 将图像转换为灰度图像, 加快检测速度
15     faceRects = classifier.detectMultiScale(gray, 1.05, 3) # 人脸检测, 返回值为矩形框的左上角坐标和长宽
16     if len(faceRects) > 0: # 如果检测到人脸
17         for faceRect in faceRects: # 遍历每一个人脸
18             x, y, w, h = faceRect # 获取矩形框的左上角坐标和长宽
19             cv2.rectangle(frame, (x - 10, y - 10), (x + w - 10, y + h - 10), color, 2) # 画出矩形框
20             cv2.imshow('frame', frame) # 显示图像
21             if cv2.waitKey(1) & 0xFF == ord('q'): # 按q键退出
22                 break
23 cap.release() # 释放摄像头
24 cv2.destroyAllWindows() # 释放窗口资源
```

分析：利用 opencv 启动笔记本上的摄像头，实时读取摄像头中的视频流，

对视频流中每一帧图像，通过加载 `haarcascade_frontalface_default.xml` 分类器文件创建一个级联分类器，读取视频片段进行灰度处理后，进行人脸检测，并返回检测到的人脸区域坐标信息，根据坐标绘制一个绿色的矩形框，并显示。

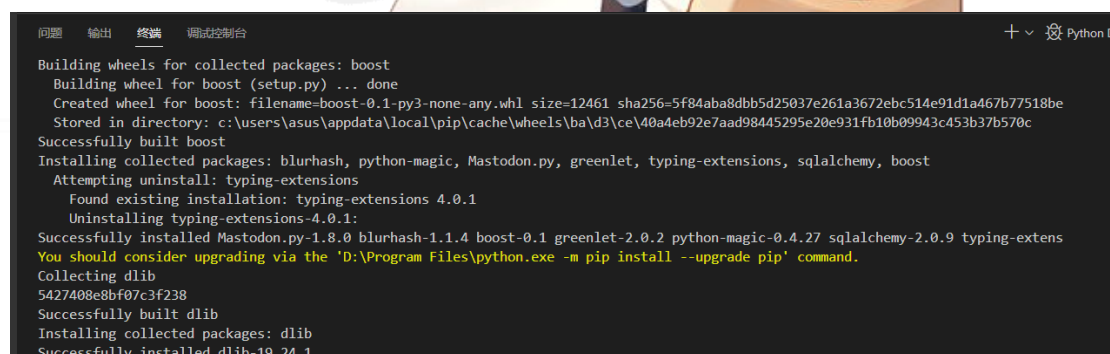
2) 使用 dlib 检测 68 个人脸特征点，代码见 task2.py

①先依次安装库

1、使用 `pip install Cmake` 安装 Cmake 库

2、使用 `pip install boost` 安装 boost 库

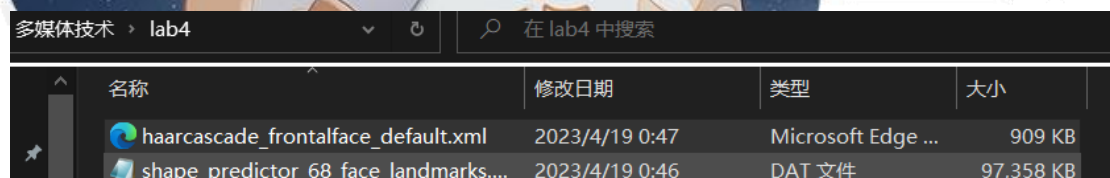
3、最后输入 `pip install dlib`



```
Building wheels for collected packages: boost
Building wheel for boost (setup.py) ... done
Created wheel for boost: filename=boost-0.1-py3-none-any.whl size=12461 sha256=5f84aba8dbb5d25037e261a3672ebc514e91d1a467b77518be
Stored in directory: c:\users\asus\appdata\local\pip\cache\wheels\ba\d3\ce\40a4eb92e7aad98445295e20e931fb10b09943c453b37b570c
Successfully built boost
Installing collected packages: blurhash, python-magic, Mastodon.py, greenlet, typing-extensions, sqlalchemy, boost
Attempting uninstall: typing-extensions
Found existing installation: typing-extensions 4.0.1
Uninstalling typing-extensions-4.0.1:
Successfully installed Mastodon.py-1.8.0 blurhash-1.1.4 boost-0.1 greenlet-2.0.2 python-magic-0.4.27 sqlalchemy-2.0.9 typing-extens
You should consider upgrading via the 'D:\Program Files\python.exe -m pip install --upgrade pip' command.
Collecting dlib
5427408e8bf07c3f238
Successfully built dlib
Installing collected packages: dlib
Successfully installed dlib-19.24.1
```

②下载使用 dlib 检测 68 个人脸特征点使用文件：

`shape_predictor_68_face_landmarks.dat`



名称	修改日期	类型	大小
haarcascade_frontalface_default.xml	2023/4/19 0:47	Microsoft Edge ...	909 KB
shape_predictor_68_face_landmarks....	2023/4/19 0:46	DAT 文件	97,358 KB

③编写代码

```
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
task1.py task2.py x
task2.py > ...
1 import cv2
2 import dlib # 人脸关键点检测
3 import numpy as np
4
5 cap = cv2.VideoCapture(0, cv2.CAP_DSHOW) # 首参数camera_id为设备摄像头的id, 一般为0, 如果有多个摄像头, 可以设置
6
7 classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml") # 加载人脸识别分类器, 这里使用的是opencv
8
9 color = (0, 225, 0) # 取255, 为绿色框
10 detector = dlib.get_frontal_face_detector() # 人脸检测器
11 predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat") # 人脸关键点检测器
12 while cap.isOpened():
13     ret, frame = cap.read() # 读取一帧数据, frame表示摄像头读取的图像矩阵mat类型
14
15     gray = cv2.cvtColor(frame,
16                             cv2.COLOR_BGR2GRAY) # 将图像转换为灰度图像, 加快检测速度
17     rects = detector(gray, 0) # 人脸检测, 返回值为矩形框的左上角坐标和长宽
18     for i in range(len(rects)): # 遍历每一个人脸
19         landmarks = np.matrix([[p.x, p.y] for p in predictor(frame, rects[i].parts())]) # 人脸关键点检测
20         for idx, point in enumerate(landmarks): # 遍历每一个关键点
21             pos = (point[0, 0], point[0, 1]) # 获取关键点坐标
22             cv2.circle(frame, pos, 2, color=(0, 255, 0)) # 画出关键点
23         font = cv2.FONT_HERSHEY_SIMPLEX # 设置字体
24     cv2.imshow('frame', frame) # 显示图像
25     if cv2.waitKey(1) & 0xFF == ord('q'): # 按q键退出
26         break
27
28 cap.release() # 释放摄像头
29 cv2.destroyAllWindows() # 释放窗口资源
```

分析: 利用 opencv 启动笔记本上的摄像头, 实时读取摄像头中的视频流, 对视频流中每一帧图像, 之后读取视频进行灰度处理后, 使用 dlib 检测 68 个人脸特征点, 并在检测到的人脸特征点画在图像帧上, 最后将视频流和人脸特征点实时显示在 opencv 窗口中

3.其他

由于之前已经完成了 vscode 的安装和 python 的配置, 便不在报告中阐述配置过程。

本次实验使用 VSCode 集成开发环境学习了 opencv 的视频捕获与人脸检测功能以及 dlib 的人脸特征点检测功能, 收获颇丰。