Project-2021 Fatigue Detection for Driving

1 最终大作业内容

本次数字图像处理及应用最终作业的课程任务为驾驶疲劳检测。具体内容包括从若干个源视频中判断：

- 眼睛是睁开还是闭合
- 是否在打电话
- 嘴巴是张开还是闭合
- 头是否在看其他地方

1.1 作业分析

视频源为 10 帧的红外视频图像，总共有四种人作为疲劳驾驶的检测对象。在部分视频中，被检测的人会戴上眼镜。人脸的位置也有不同的变化，有的人脸是正对镜头，有的则是侧身，并且视频中一般包含被检测者的手臂。

图像画面呈现灰度图，没有明显的色彩，因此通过肤色来定位人脸不可行。想要通过边缘的方法来找到人眼，嘴巴等，受视频质量的影响，边缘检测并不稳定，不同帧的检测效果不同，存在冗余和丢失的情况。通过阈值的方法找关键部位也不容易，灰度图像中有太多阈值相似的干扰，眼睛和眉毛在阈值选取不恰当的情况下很容易混淆。因为源视频中人的位置有变化，体型、人脸的形状和方向也有变化，并且眼睛这一因素对眼睛的定位会产生很大的影响，有的图中人眼与上眼镜框重合。

尝试使用课程中所学的图像处理方法没有很好的解决办法（**所尝试方法具体在 3.6 节中详细介绍**），而不能使用深度学习的方法完成，查阅资料找到一种基于梯度直方图和支持向量机的人脸检测方法，通过梯度直方图（HOG）提取特征，再通过支持向量机（SVM）对特征进行分类，完成人脸特征的检测，从而找到人脸。

1.2 dlib 人脸检测

Python 中的 dlib 是一个强大的机器学习库，其中包含有效果很好的人脸关键点检测模型。其中人脸 5 点和 68 个关键点的已经有训练好的模型。

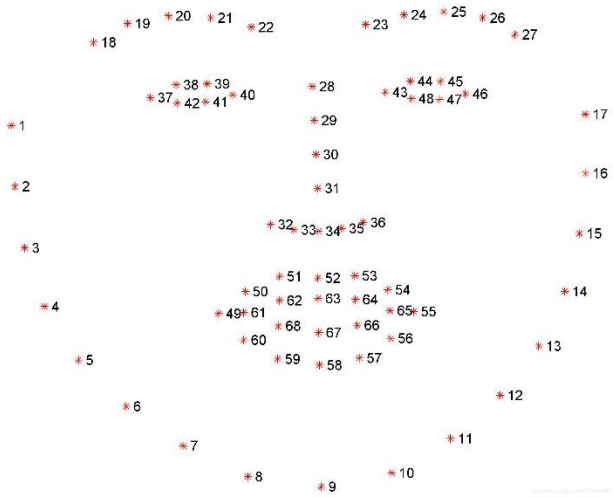

图 1. 1 人脸关键点

如图 1. 1 为人脸 68 个关键点的图示：1-17 点为脸颊；18-22 为右眉毛；23-27 为左眉毛；28-31 为鼻梁；32-36 为鼻尖；37-42 为右眼；43-48 为左眼；49-60 为外嘴唇；61-68 为内嘴唇。

在本次大作业中，将借助训练好的人脸模型文件进行疲劳驾驶行为检测。

2 思路及相关程序

对于四种检测问题，都是在提取到人脸 68 个关键点的前提下进行进一步分析的。因此，检测效果好的前提是人脸识别成功。

通过以下语句生成人脸关键点检测器并加载对应的关键点模型：

```
1. detect = dlib.get_frontal_face_detector()
2. predict = dlib.shape_predictor("models/shape_predictor_68_face_landmarks.dat")
```

通过交互式的文件选择模块，选择需要用作对应检测的视频文件，通过 opencv 库中的 VideoCapture 读取视频，并通过 VideoWriter 准备输出视频。

```
1. file_path = ''
2. while file_path =='':
3.     file_path = filedialog.askopenfilenames(title='请选择一个视频',initialdir='resources/calling/')
4. choose_file = file_path[0]
5. print(choose_file[-6:])
6.
7. cap=cv2.VideoCapture(choose_file) #视频路径
8. # cap=cv2.VideoCapture('resources/eye/eye_closed_21.avi') #视频路径
9.
```

```
# 准备输出视频流
fps =cap.get(cv2.CAP_PROP_FPS)
size = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)), int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
# width = 960
# height = int((int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) / int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))) * width)
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
out = cv2.VideoWriter('../results/calling/Phoning_res_'+choose_file[-6:]+'',fourcc, fps, size)
```

视频播放采用循环语句,通过 cap.read()读取每一帧的图像，按照原来的帧率显示，在关闭图像显示窗口之后输出带检测结果的视频，程序停止运行。

对于四种具体检测情况，采取了不同的判别方式。

2.1 眼睛闭合或睁开

在眼睛睁开或闭合部分增加一个被 EAR 系数。

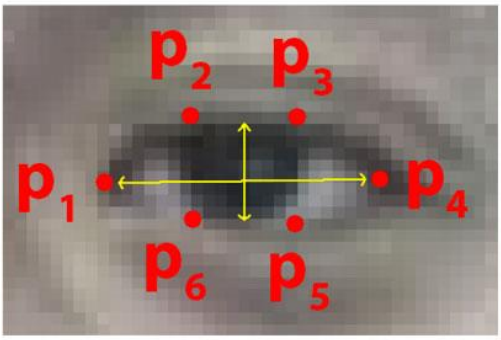

图 2.1 人眼关键点

如图 2.1 所示人眼部分的关键点共有 6 个，顺次连接 6 个点能够形成一个六边形，人眼张开使，六边形的上下边之间的距离相对于左右两点的距离远；而眼睛闭合时的距离大，因此，可以根据这个比例来判断眼睛睁开还是闭合，定义：

$$EAR = \frac{|p_2 p_6| + |p_3 p_5|}{2|p_1 p_4|}$$

根据从视频中测试得到的结果估计判断眼睛闭合与否的阈值大小。

```
def eyeAspectRatio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
```

最后，在输出视频中，找到人脸的轮廓，并根据眼睛的位置画一个矩形框，并将判断的结果显示出来。

2.2 是否在打电话

判断是否在打电话和人脸也有关系，从源视频中能够发现，被检测对象在接电话时，都会将手机放在左侧或者右侧的耳边，同时，原来这个耳边的区域在车的背景下，整体偏暗，而将手放在耳边，这个区域由于出现了手，整体会变亮。根据这一特点，我们根据脸两侧的一个矩形区域，计算其中的平均灰度值，根据这个平均值判断是否在接打电话。

```
region1  = gray[top1:bottom1, left1:right1]
region2  = gray[top2:bottom2, left2:right2]

mean_left = np.sum(region1) / (right1 - left1) / (bottom1 - top1)
mean_right = np.sum(region2) / (right2 - left2) / (bottom2 - top2)
# 判断打电话
if mean_left > thresh:
    cv2.rectangle(frame, (left1, top1), (right1, bottom1), (0, 255, 0), 3)
    cv2.putText(frame, "calling", (left1, top1-15),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
elif mean_right > thresh:
    cv2.rectangle(frame, (left2, top2), (right2, bottom2), (0, 255, 0), 3)
    cv2.putText(frame, "calling", (left2, top2-15),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
else:
    cv2.putText(frame, "not calling", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
```

判断平均灰度值的阈值根据视频检测结果的具体表现确定。

2.3 嘴巴是否张开

嘴巴的张开闭合与眼睛的睁开闭合有相同的原理，因此，同样为嘴巴设置一个 MAR 系数，选取嘴巴关键点中对应合适的点（对应人脸 68 关键点中的 49、51、53、55、57、59），定义：

$$MAR = \frac{|p_2 p_6| + |p_3 p_5|}{2|p_1 p_4|}$$

根据视频中的测试效果决定阈值设置的大小。

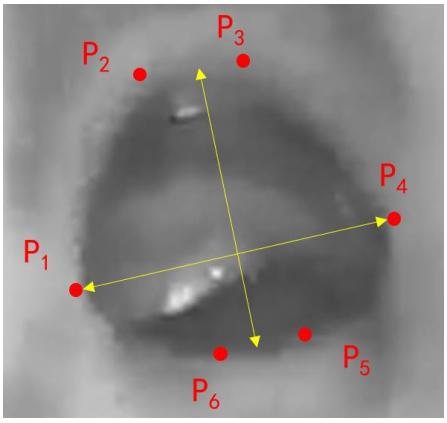

图 2.2 人嘴关键点

```
def mouthAspectRatio(mouth):
    A = distance.euclidean(mouth[2], mouth[10])
    B = distance.euclidean(mouth[4], mouth[8])
    C = distance.euclidean(mouth[0], mouth[6])
    mar = (A + B) / (2.0 * C)
    return mar
```

2.4 是否四处张望

源视频是车内部红外设备采集到的图像，所以假设视频中是一定存在人头的，但是具体在看向正前方还是在四处张望不清楚。而通过支持梯度直方图和向量机得到的人脸关键点很难检测到侧脸，尤其是侧脸幅度大的情况。因此，判断被检测对象是否在四处张望的一种办法就是判断视频帧中是否存在人脸。当检测到人脸时，判断没有四处看；而没有检测到人脸的时候，就判断为在四处看。

在程序中通过检测器 detector 将检测到的人脸存放到 faces 变量中，如果没有检测到人脸，则 faces 的长度为 0，作一个简单的判断可以用来检测是否四处张望。

```
faces = detect(gray, 0)

if len(faces) == 0:
    cv2.putText(frame, "looking around", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
else:
    cv2.putText(frame, "not looking around", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
out.write(frame)
cv2.imshow("Frame", frame)
```

3 结果

3.1 眼睛闭合或睁开

运行 Drowsiness_Detection_eye.py，得到部分测试结果如图 3.1 所示：

图 3. 1 人眼睁开闭合检测结果

从结果图 3. 1 中可以看到，多测测试选择合适的阈值之后，能够得到较好的人眼状态检测效果，进行了正确的判断。最后得到的 EAR 系数的一个较合适的值为 0.32。

3.2 是否在打电话

运行 Drowsiness_Detection_calling.py，得到部分测试结果如图 3. 2 所示：

图 3. 2 是否接打电话检测结果

从检测结果中我们能够发现是否接打电话的行为也能够被正确检测出来，当接打电话时，会在解答电话的地方框出一个区域并显示正在打电话；而不接电话则在

画面的左上角显示：not calling。经过测试得到的合适平均灰度值为 75。左手接打电话的结果如图 3. 3 所示：

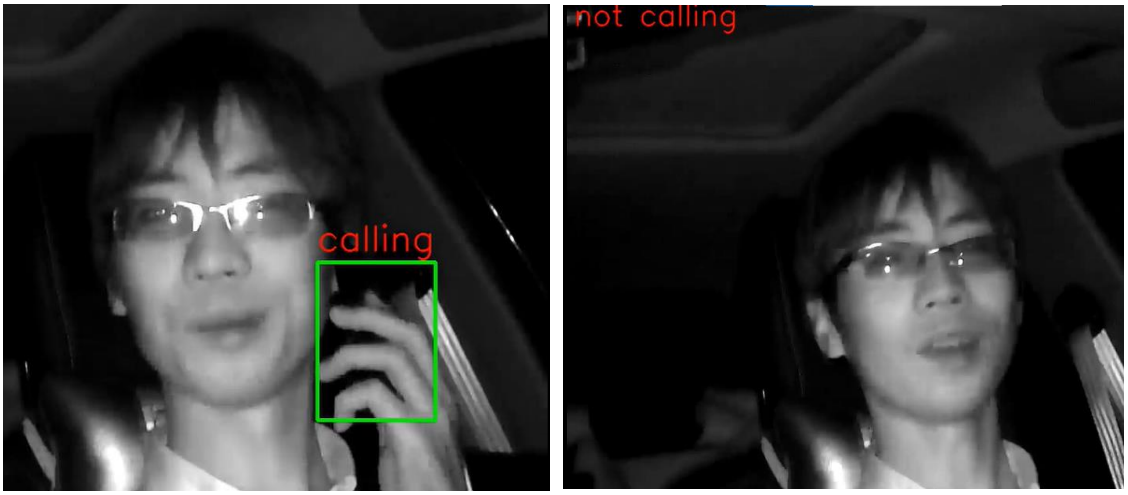

图 3. 3 是否接打电话检测结果（左手）

3.3 嘴巴是否张开

运行 Drowsiness_Detection_yawn.py，得到部分测试结果如图 3. 4 所示：

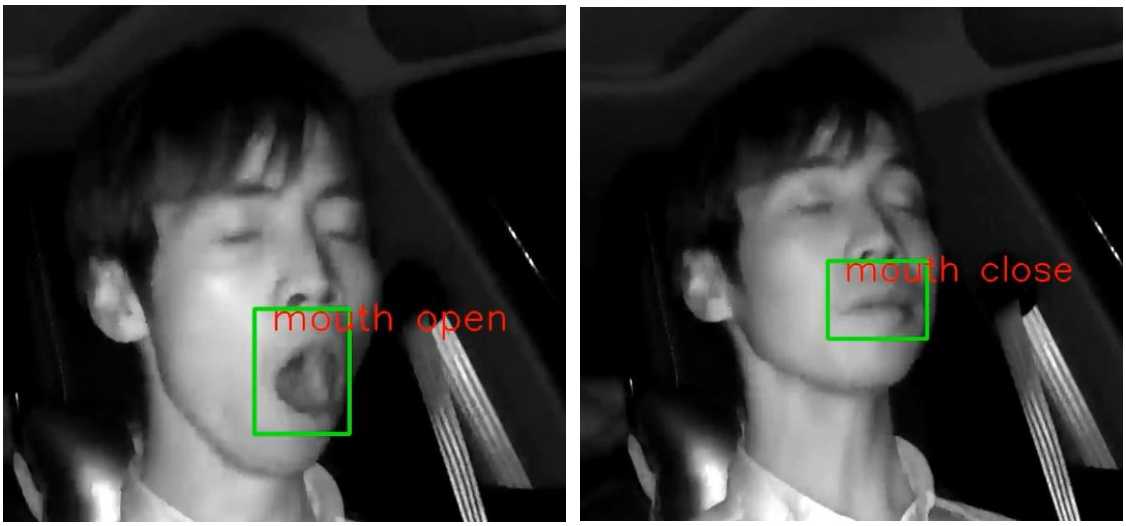

图 3. 4 嘴巴是否张开检测结果

检测的结果也与眼睛是否睁开类似，检测到人嘴巴的位置，框出，计算 MAR 系数，并将判断的结果显示在框附近。从结果看该方法也能够正确判断是否张嘴。经过测试得到的较好的 MAR 系数阈值为 0.54。可以发现这个阈值比眼睛的 EAR 阈值要更大一些。这是因为选择的是外嘴唇，若选择内嘴唇最为判断系数阈值将会接近甚至比 EAR 系数更小。

3.4 是否四处张望

运行 Drowsiness_Detection_looking.py，得到部分测试结果图 3. 5 所示：

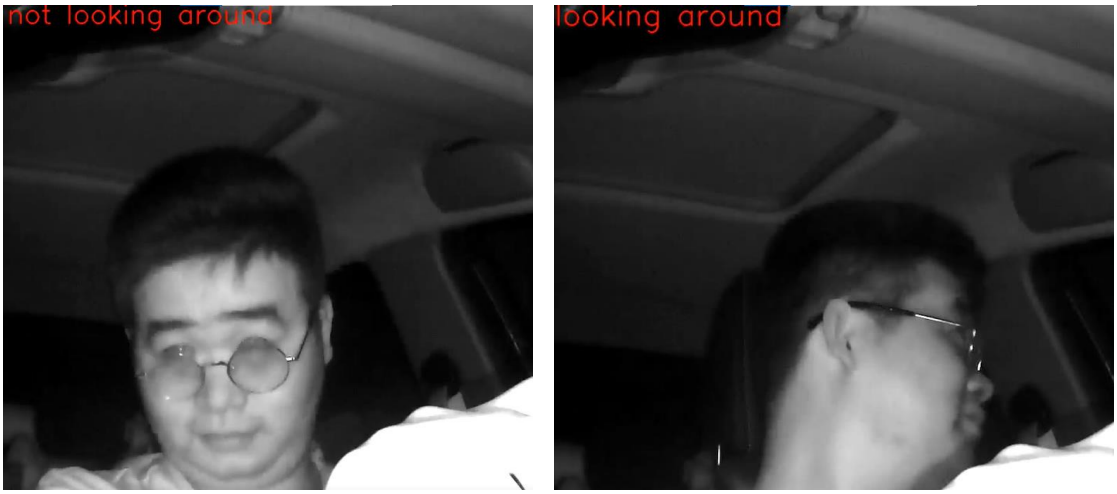

图 3. 5 是否四处张望检测结果

从检测结果可以看出，程序能够成功判断被检测对象是否在四处看，并将判断的结果显示在了图片左上角。

3.5 问题及优化

四处张望优化

第一个问题是在是否四处张望的判断函数中，最开始是通过能不能检测到人脸进行判断，这个条件在大部分情况下都是成立的，但是视频镜头的位置位于对象的右前方，当对象向左看时，是检测基本不到人脸的；但是对象向右看时，一定角度的右转头可能也能够检测到人脸，这个时候判断就出错了（图 3. 6 左）。

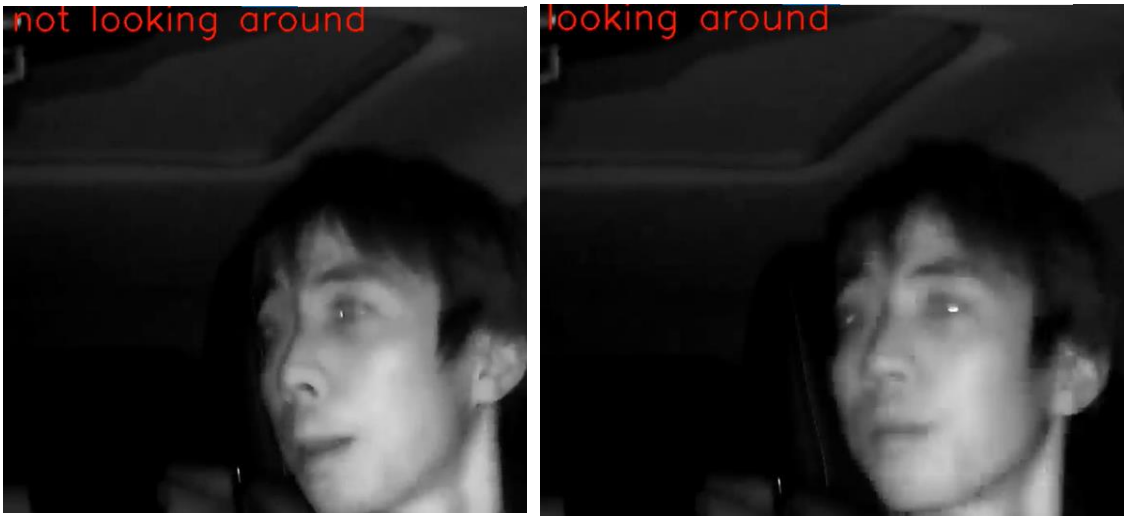

图 3. 6 优化前后四处张望检测效果

对于这个问题，可以发现一个规律：向右转头的时候检测到的人脸左眼到鼻梁的水平距离一定是大于右眼到鼻梁的水平距离的。因此，可以通过设定一个阈值，当两个距离的比值小于这个阈值，对象正在向右看，也判断为在四处张望。原处理的效果和优化后的处理效果对比如图 3. 6 所示（在此阈值设置为 0.7）。

```
dx_r = abs(face_68_point[39][0] - face_68_point[27][0])
dx_l = abs(face_68_point[42][0] - face_68_point[27][0])
if dx_r/dx_l < T:
    cv2.putText(frame, "looking around", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
else:
    cv2.putText(frame, "not looking around", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
```

人脸漏检

在测试的过程中也发现一个问题：直接输入源图像进行检测，有比较大的概率会出现检测不到人脸的情况（图 3. 7）这种情况对于检测的结果有很大影响。

图 3. 7 检测不到人脸没有显示结果

对这种情况，经过观察分析，认为输出的源图像画面较暗，图像质量不佳，特别是被检测对象在佩戴眼睛的时候，能够大幅削减眼睛周围的特征，对检测产生较大的干扰。同时由于是红外图像，与该检测模型训练的样本有部分差异，所以不能够很好地检测到人脸，对此，进行中值滤波并采用局部直方图进行增强。

中值滤波能够滤除图像中的部分噪声，同时相比均值滤波对边缘的破坏没有那么严重。局部直方图与直方图均衡不同，不是在全图像范围内使各种灰度占比均衡，而是在一个个小窗口内进行部分均衡，能够增强细节，提高对比度，提升检测兄效果。

图 3.8 直方图均衡处理结果

图 3.9 局部直方图处理结果

从图 3.8、图 3.9 的对比结果可以看出，在这种情况下局部直方图的增强效果要由于直方图均衡。进行滤波后再通过局部直方图增强后的结果对比如图 3.10 所示：

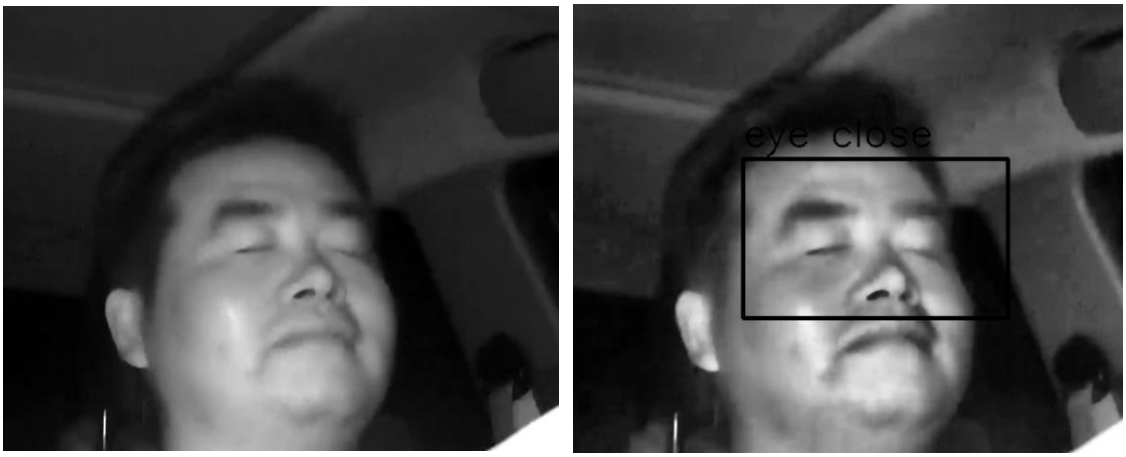

图 3.10 进行直方图均衡前后结果对比

从图 3.10 中可以看出，检测不到人脸的情况得到了一定程度的缓解，能够进行图像增强，使得人脸更容易被检测到。中值滤波的参数选择为 3*3 的滤波窗口大小，而局部直方图窗口大小为 8*8，最大对比度限制为 2，以免对比度过大反而减弱增强效果。图像预处理程序：

```
# 图像预处理
# 中值滤波
gray = cv2.medianBlur(gray, 3)

# 高斯滤波
gray = cv2.GaussianBlur(gray,(3, 3),0)

```

```
# 局部直方图
calhe = cv2.createCLAHE(clipLimit=2, tileGridSize=(8, 8))
gray = calhe.apply(gray)
```

进行滤波和局部直方图增强后，检测过程中仍存在部分检测不到人脸的情况，在戴眼镜的情况下人眼的检测效果有时也很差，人眼位置容易判断失误，与眼镜边框及眉毛混淆，使得人眼是否闭合的检测结果出现问题。车内右侧的白色安全带灰度值较高，有时候会对判断是否接打电话产生影响，将未接打电话误判为打电话。

这些问题经过多次尝试其他方式的增强（伽马变换、锐化等），以及优化措施都没有明显的提升效果，后面仍没有合适的优化方案，希望能够得到解惑。最终综合检测效果，平均能够正常检测并且得到正确结果的帧数大概占 60%~70%。

图 3. 11 检测失败情况

3.6 尝试及失败方案

首先尝试用经典 Canny 边缘检测方法对一些视频进行分析，观察边缘效果。

图 3. 12 Canny 边缘提取结果

图 3. 13 形态学结合水平边缘处理结果

如图 3. 12 所示，参数设置为（13，63）的 canny 算法检测图像边缘的结果。

可以看到，眼镜虽然出现了大致的轮廓，但是因为边界不明显以及眼镜的影响，眼睛有大致的轮廓但是不清晰，容易丢失。在人脸部位也有鼻子轮廓，眉毛等干扰，通过边缘定位眼睛并判断睁闭眼没有思路。

考虑结合图像形态学操作的知识。视频中的人脸没有很大的倾斜，这意味着，人脸眼睛闭上的时候近似一条水平的线。因此，考虑通过以下步骤筛选出人眼的大致位置：1. 通过水平方向的梯度算子得到一个放大水平边缘特征的图；2. 结合形态学开、闭操作删除图像中的噪声，同时连接断开的边缘；3. 构造一个扁平的叶子状的模板（近似人眼形状），通过击中击不中变换对 2 中处理后的图像进行处理。得到的结果如图 3. 13 所示。形态学处理后的图像中的杂点确实去掉了很多，留下了眼睛附近的部分点，但是如眼镜这种水平方向的边缘响应也很强烈，同样被保留了一部分下来，难以通过一些手段将人眼位置确定，这个角度的思路也断了。

注意到，图像中，人脸有一个特点：在图片中，眼睛等器官相对于大面积的人脸是颜色较深的，可以想到通过用 otsu 最大类间方差的方法分割图像，可以期待得到一个中间包含孔洞的人脸分割结果。对图像先进行中值滤波去除一定的噪声，再通过 otsu 法对图像阈值分割，分割后的结果如图 3. 14 所示。

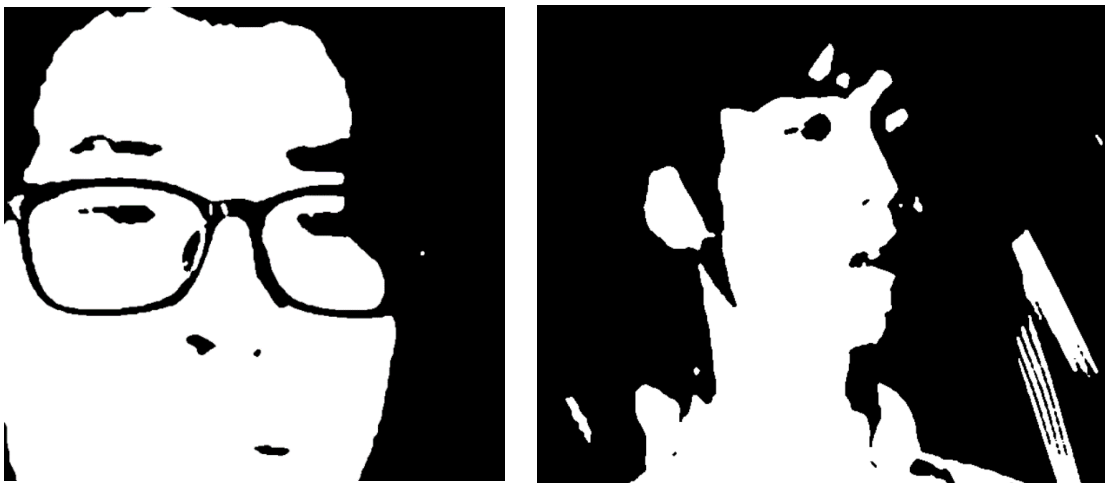

图 3. 14 otsu 阈值分割结果

从经过 otsu 阈值分割的结果我们可以发现，确实出现了类似于在一个大区域中有一些孔洞的情况，或许通过孔洞填充的方式，并和源图像进行一个差的操作能够定位到这些孔洞。但是可以看到在源图像中，包含了很多鼻孔，眉毛等形成的孔洞，这些孔洞有的面积与形状与眼镜形成的孔洞类似，这些孔洞对于眼睛的提取会造成很大的干扰，没有找到一个泛用的提取方法，进一步判断睁闭眼也没有思路。

4 总结

通过这次大作业，首先对连续情况的图像处理——视频图像处理有了更深的了解和应用实践，能够学习运用灰度、阈值、滤波和图像增强的方法得到想要的效果。但是这次作业的跳跃感较大，学习过程中都是针对单张图片进行处理，但是连续的图像变化大，在完成大作业的过程中也尝试过运用边缘、阈值、滤波的方法找解决方案，但是视频源的变化很大，很难针对性地找到一个泛用的方法来解决疲劳检测的问题，尝试的几种方法也都没有想到解决途径。源图像中人物、眼镜、镜头角度的变化都使得这个问题的处理变得很复杂，一时间很难联想到如何运用学习的知识来解决这个问题，所以最后通过网络上的资料选择了已有的模型辅助解决，课程所学内容体现不多。学习的知识和在大作业中的应用割裂感很强，一方面自己对所学知识的运用也还是不熟练，对方法的了解不够深刻，创新的思维不够活跃，这些也是今后学习中需要加强的能力。