2022年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛

2022 Intel Cup Undergraduate Electronic Design Contest

- Embedded System Design Invitational Contest

**作品设计报告**

**Final Report**



报告题目：安车行——车载安全驾驶系统

学生姓名: 宁智伟 杨珍妮 李沂坤

指导教师: 张育林

参赛学校: 西安交通大学

**2022年英特尔杯大学生电子设计****竞赛嵌入式系统专题邀请赛**

**参赛作品原创性声明**

本人郑重声明：所呈交的参赛作品报告，是本人和队友独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果，不侵犯任何第三方的知识产权或其他权利。本人完全意识到本声明的法律结果由本人承担。

参赛队员签名：

日期： 年 月 日

**安车行——车载安全驾驶系统**

摘要

本项目基于GNS-V40硬件平台，应用Intel公司的OpenVINOTM工具套件，打造一款服务于司机安全驾驶的智能监测系统。系统通过车载摄像头进行人脸检测，通过深度学习算法进行人脸识别，并利用人眼特征点进行疲劳程度的判断。使用OpenPose模型进行人体关键点的提取与危险行为检测。使用OpenVINO框架进行路况检测的加速。同时，系统具有多模态人机交互的功能。综合而言，本系统关注当下的重点问题，切实保护驾驶员与乘客的人身安全，具有很好的应用前景与市场需求。

关键词：计算机视觉，人脸检测，深度学习，安全驾驶，智能交互

**DRIVE SECURER —— Vehicle safety driving system**

**ABSTRACT**

This project is based on GNS-V40 hardware platform and uses OpenVINOTM tool suite of Intel company to create an intelligent monitoring system for safe driving. The system performs face recognition and detection by vehicle camera, face recognition by deep learning algorithm, and judgment of fatigue degree by human eye feature points. Openpose model was used to extract key points of human body and detect dangerous behaviors. The OpenVINO framework is used to accelerate road detection, and it has the function of multi-mode human-computer interaction. In general, this system focuses on the current key issues, effectively protect the safety of drivers and passengers, has a good application prospect and market demand.

**Key words:** computer vision, fall detection, voice interaction, health monitoring, companion

目 录

1. 绪论----------------------------------------------------------------------------------------------------1
   1. 项目背景---------------------------------------------------------------------------1

1.1.1 机动车使用量增多-------------------------------------------------------------------------1

1.1.2 交通事故频发--------------------------------------------------------------------------1

1.1.3 更加便捷的人机交互需求--------------------------------------------------------------1

* 1. 国内外研究现状------------------------------------------------------------------------2

1.2.1 国外研究现状--------------------------------------------------------------------------2

1.2.2 国内研究现状--------------------------------------------------------------------------3

* 1. 研究目的及主要内容------------------------------------------------------------------------3

1.3.1 研究目的--------------------------------------------------------------------------3

1.3.2 主要内容-------------------------------------------------------------------------3

* 1. 研究方法与技术路线------------------------------------------------------------------------4

1. 系统概述-------------------------------------------------------------------------------------5
   1. 需求分析---------------------------------------------------------------------------------------5
   2. 设计原则-----------------------------------------------------------------------------5
   3. 功能与指标------------------------------------------------------------------------5

2.3.1 疲劳检测--------------------------------------------------------------------------5

2.3.2 酒驾检测-------------------------------------------------------------------------5

2.3.3 吸烟检测--------------------------------------------------------------------------5

2.3.4 拨打电话监测----------------------------------------------------------------------6

2.3.5 路况检测-----------------------------------------------------------------------6

2.3.6 娱乐放松--------------------------------------------------------------------------6

2.4系统组成------------------------------------------------------------------------------6

2.5硬件框图------------------------------------------------------------------------------6

2.6软件流程-------------------------------------------------------------------------------7

1. 系统原理及方案--------------------------------------------------------------------9

3.1疲劳检测-------------------------------------------------------------------------------9

3.1.1 研究综述--------------------------------------------------------------------------9

3.1.2 Dlib原理--------------------------------------------------------------------11

3.1.3 疲劳检测算法分析-----------------------------------------------------------------11

3.2危险行为检测------------------------------------------------------------------------------15

3.2.1 Openpose原理-------------------------------------------------------------------15

3.2.2 基于人体关键点的危险行为检测算法-------------------------------------------16

3.3情绪检测---------------------------------------------------------------------------------17

3.4路况检测------------------------------------------------------------------------------18

3.4.1 Mobilenetv2模型原理---------------------------------------------------18

3.4.2 单目测距原理---------------------------------------------------------------------20

3.5酒精检测-----------------------------------------------------------------------------------------20

3.6人机交互与多媒体娱乐----------------------------------------------------------------------------21

3.6.1 基于深度学习的语音识别----------------------------------------------------------21

3.6.2基于百度语音API的语音识别与交互-------------------------------------------------22

3.6.3 基于语音模块的语音识别-----------------------------------------------------------------22

3.6.4 UI界面----------------------------------------------------------------------------------23

3.6.5 音乐播放----------------------------------------------------------------------------------------24

1. 系统测试-------------------------------------------------------------------------------------25

4.1系统测试方案-----------------------------------------------------------------------------25

4.2测试设备--------------------------------------------------------------------------------------25

4.3疲劳检测测试---------------------------------------------------------------------------25

4.4情绪检测测试------------------------------------------------------------------------------26

4.5危险行为检测测试----------------------------------------------------------------------------26

4.6路况检测测试----------------------------------------------------------------------------26

4.7智能交互检测测试----------------------------------------------------------------------------27

4.8整体性测试----------------------------------------------------------------------------27

4.9结果分析-------------------------------------------------------------------------------27

1. 总结与展望---------------------------------------------------------------------------------------28

5.1系统特色-----------------------------------------------------------------------------28

5.2前景展望------------------------------------------------------------------------------28

参考文献----------------------------------------------------------------------------------------------------------30

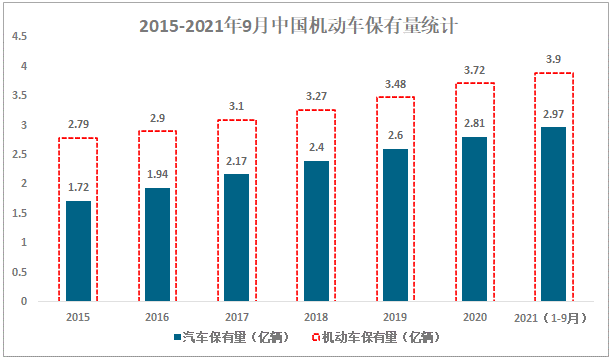
附录（部分程序源代码）-------------------------------------------------------------------------------------31

**第一章 绪论**

1.1 项目背景

1.1.1 机动车使用量增多

随着我国经济的不断发展，人们对于机动车的使用日益增多。根据历年的国民经济和社会发展统计公报数据显示，在2021年末，我国的民用汽车保有量高达30151万辆，比上年末增加2064万辆，增长率达6.85%，由此引发出了越来越多的道路交通安全问题[1]。



**图1-1 近几年我国机动车数据变化**

**（数据来源：国家统计局）**

1.1.2 交通事故频发

根据发生的交通事故的调查数据显示，车辆驾驶员在驾驶过程中存在如疲劳驾驶、驾驶过程中用手机接打电话等一系列问题。

上述一系列不规范行为在驾驶过程中存在着许多风险，常常会发生意外，其中以疲劳驾驶、抽烟、打电话等情况最为普遍。

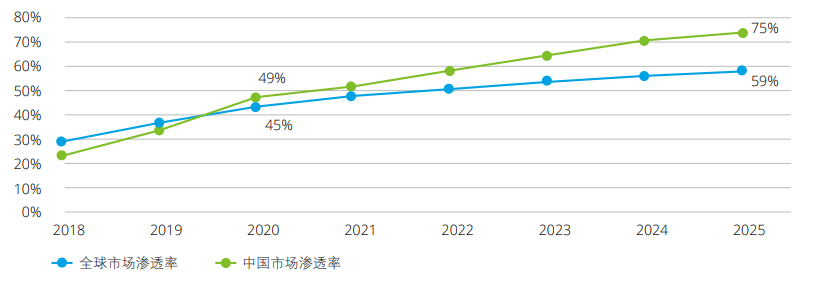
根据公路交通管理局调查数据显示，由疲劳驾驶导致的交通事故死亡人数占交通事故总死亡人数的21%，疲劳驾驶，成为了交通事故产生的最主要原因之一，给家庭和社会带来了巨大的损失。如何有效检测疲劳驾驶，避免不必要的交通事故已经成为机动车行业亟待解决的问题[2]。

在驾驶过程中，驾驶员出现接打电话，抽烟等行为，不仅会使驾驶员单手甚至双手脱离方向盘，还会分散驾驶员的注意力，影响驾驶员对环境的判断，降低了反应速度，容易引发交通事故[3]，而目前尚缺乏一系列及时有效的措施对这些危险行为进行监管，由此可见，不规范驾驶行为存在诸多风险。

在疲劳驾驶占据交通事故发生原因超过20%的情况下，交通法规明确规定禁止疲劳驾驶和驾驶车辆中拨打电话等危险行为，交警通常通过行车记录仪和道路监控录像在事故发生后判断驾驶员是否疲劳驾驶等。但是，这种问责总是发生在事故发生后，通过问责和处分没有从根本上解决问题，这些追溯都相对滞后，所以很有必要针对疲劳驾驶、危险行为进行实时的检测，同时通过语音交互，打造一款聚焦于驾驶员安全驾驶以及操作便捷的多模态人机交互安全监测系统。

1.1.3 更加便捷的人车交互需求

在车载场景中，最为突出的特点是驾驶员的注意力会被驾驶过程占用，从而为屏幕操作的人机交互方式带来不便。在此情景下，车载语音助手变的尤为重要。同时，智能语音导航也成为各大厂商的青睐，如图1-2所示。语音导航、语音控制、语音提醒，车载信息娱乐系统等车载交互系统，解放驾驶员双手的同时，也为驾驶员和乘客带来了安全便捷出行和娱乐享受的驾驶体验[4]。



**图1-2 语音操作系统在智能车市场渗透率**

在驾驶过程中，大部分人普遍关注驾驶过程中的安全情况，却忽视了在车载场景中，便捷交互的需求。当前驾驶系统的交互方式仍旧以传统的屏幕操作为主，语音交互，智能互联的方式正逐渐被提出[5]。

1.2国内外研究现状

随着人工智能、大数据、物联网技术的发展，国内外安全驾驶产业取得许多新的突破，涌现出了一批新型驾驶员监测产品，了解国内外相关研究背景与现状能为我们项目的开展提供帮助。

目前国内外对预防驾驶员疲劳驾驶研究众多，一般分为三种监测技术：第一种监测技术主要通过监视方向盘与踏板运动实现，该技术根据方向盘运动规律与踏板使用情况以确定驾驶员的疲劳程度。第二类技术集中于生物电信号，例如 EEG(脑电图)，ECG(心电图)和 EOG(眼电图) [6]。但是，以上两种监测技术具有一定的局限性。第一类技术只能在特定的驾驶条件下使用。第二类技术由于需要在驾驶员身上佩戴各种信号测量工具，使驾驶员有不适感，所以难以实际应用。因此，基于计算机视觉的驾驶员疲劳监测技术变得越来越流行。

1.2.1 国外研究现状

Dwivedi 等人采用浅层 CNN 进行驾驶员疲劳检测，准确率可以达到 78％。S.Park 等人提出了一种使用三层网络的新架构，在第一层网络中，图像特征是通过使用 AlexNet 来学习的，AlexNet 由 5 个 CNN 和 3 个 FC 层组成；在第二层网络中，通过 16 层 VGG-Face Net 提取面部特征；在第三层网络中，通过使用FlowImageNet 来提取行为特征，该架构能够以 73％的检测精度对驾驶员行为进行检测。Wang Rongben 等人提出了一种通过安装在车内的摄像头来检测驾驶员是否分心或疲劳的系统，摄像头可以直接警示驾驶员[7]。

许多汽车公司与第三方公司开始设计、研发与使用监测驾驶员状态的产品。2016 年日产汽车的驾驶员警报系统会跟踪驾驶员的转向模式，一旦检测到任何异常偏差，就会产生警告信号。日产汽车的驾驶员警报系统通过建立一个基线来适应驾驶员的行为，并对转向校正误差进行了连续的统计分析，以检测是否偏离了基线。一旦系统检测到驾驶员疲劳驾驶，控制台上将显示一个警告标志。大众汽车的休息辅助系统会根据方向盘运动规律与踏板使用情况来判断驾驶员的疲劳程度，一旦系统检测到驾驶员疲劳驾驶，车辆就会以方向盘振动、声音信号和视觉消息的形式警告驾驶员。睡眠诊断私人有限公司设计了用于驾驶员疲劳监测的产品 OPTALERT。OPTALERT 利用无线眼镜监测瞳孔和眼睑来判断驾驶员的疲劳状态，目前 OPTALERT 产品已经被部署在澳大利亚的商用车辆上[8]。

1.2.2 国内研究现状

当前，全球疫情防控工作还在紧张的进行中，在重点公共防疫场所（公交车）都需要对人员佩戴口罩行为进行监测。优视智联在疫情期间推出了全新的“口罩佩戴检测”智能产品：A121-F7 与 A242-F7。该产品采用深度智能人脸算法，提供多种包括戴口罩识别在内的 AI 行为分析检测功能。该产品内置智能模块，可以完成自动识别预警。目前该产品已经在济南、深圳等城市的公共场所完成部署。云天励飞基于人工智能技术，也推出了“口罩佩戴检测”的疫情防控解决方案[9]。目前，该方案已经在深圳、杭州、贵阳、成都、济南、东莞等城市的公共场所完成部署，积极服务各地防疫工作。耿磊等人通过红外视频采集系统分析驾驶员面部信息来检测疲劳状态，首先结合 Adaboost 与和 KCF（核相关滤波器）算法对视频帧进行人脸检测，然后在此基础上进行面部关键点定位，从而实现对眼部和嘴部的定位并对其状态进行区分判断，最后根据疲劳参数进行最终的疲劳驾驶判定[10]。另外，还有一些穿戴式安全驾驶监测系统，均对车载智能安全驾驶系统功能设置提供了一定的参考。

通过对国内外产品的调研可以发现，可以发现随着人工智能、边缘计算、深度学习等技术的不断发展，安全驾驶检测的相关研究及其产品正从最初的接触式检测向智能化和便捷化的非接触式检测进行转变，且目前的安全驾驶辅助产品在驾驶环境下对智能交互、功能多样性缺乏关注，对安全检测与人车交互结合的研究不足，在驾驶环境中，为驾驶员提供的便捷性有待提高。

1.3研究目的及主要内容

1.3.1 研究目的

基于以上研究背景，本文拟基于计算机视觉和语音交互技术，借助Intel平台的高处理能力，研发一种聚焦于驾驶员安全驾驶以及操作便捷的多模态人机交互安全监测系统，用于检测驾驶员的危险行为以及车外路况等。具体而言，本项目作品将围绕以下目标进行设计：

（1）了解交通事故频发原因，挖掘辅助驾驶员安全驾驶的核心需求；

（2）构建基于计算机视觉的车载安全驾驶系统，通过车内摄像头和车外行车记录仪的视频输入，在边缘侧对驾驶员危险行为和车外行车距离过近进行识别和报警；

（3）研究交互设计的方法理论与设计理念，提出针对驾驶员驾驶过程中的交互设计方法和原则，并设计智能交互方案;

1.3.2 主要内容

本文主要研究一种基于计算机视觉和语音交互的模态人机交互安全监测系统，对驾驶员的危险行为和车外路况进行检测。本文所做的主要工作如下：

1. 对目前国内外相关车载安全驾驶产品进行分析，提炼驾驶员安全驾驶的需求和现实痛点，定义系统功能；
2. 利用计算机视觉，分析当前国内外疲劳检测研究进展，基于YOLO4的深度学习架构，对人脸的特征点进行提取，计算EAR值来判断司机是否疲劳；
3. 对人体姿态及其动作进行了研究，结合边缘计算发展趋势和GNS-V40硬件平台强大的视觉处理能力，采用基于OpenPose提取人体关键点的算法，实现驾驶过程中系统对驾驶员危险行为的检测和报警；
4. 研究现有的车外行人检测模型，利用Mobilenetv2模型对驾驶途中车外道路上的行人、机动车、非机动车进行检测；
5. 分析交互技术相关理论和设计要点，开发对驾驶员友好的交互方式和交互界面；
6. 采用酒精传感器进行酒精检测，检测驾驶员是否酒驾；

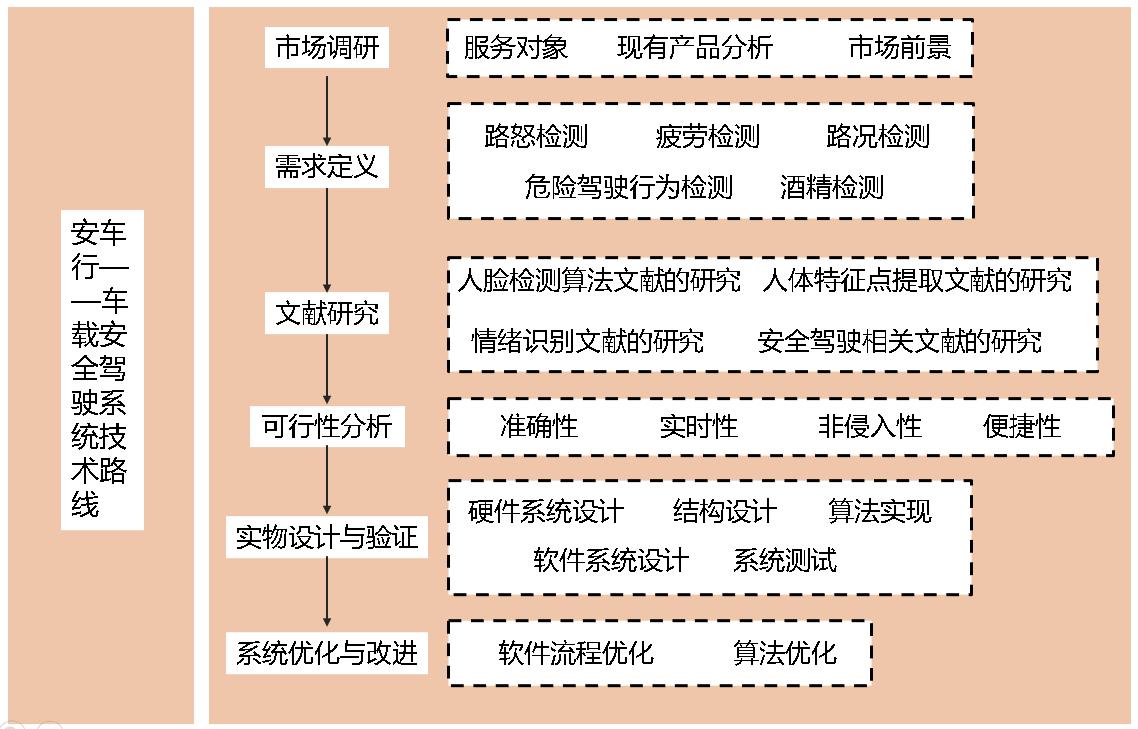
1.4研究方法与技术路线

本文的研究方法主要有：

（1）算法原理分析；发掘国内外关于安全驾驶、疲劳检测、路况检测等相关算法的原理，整理分析不同的原理，梳理安全驾驶的体系结构与研究框架，发现研究难点与不足；

（2）实物设计与对比分析：寻找合适的模型，选择合适的指标评价模型性能，基于模型进行改进和优化，实现驾驶过程中疲劳、危险行为、车外路况的检测。基于GNS-V40硬件平台进行嵌入式系统设计与开发，并与相关研究成果进行比较，分析优劣，不断加以改进[11]。

本文研究的技术路线如图1-3所示。



**图1-3 技术路线**

1. 系统概述

2.1 需求分析

通过项目前期调研和背景分析，针对当下机动车驾驶员出现的一些危险驾驶行为与交通事故起因，我们总结车载安全驾驶系统的功能需求如下：

（1）疲劳检测：系统通过对摄像头采集到的图像进行处理，当系统监测到驾驶员在驾驶过程中出现疲劳情况时，将进行相应的警报，并提醒司机。

（2）危险行为检测：针对司机的一些不安全驾驶行为，例如玩手机，抽烟等，本系统将会进行相应的提示，规范司机的驾驶行为，对降低安全隐患，改变司机的驾驶陋习起到促进作用。

（3）路况检测：结合智能驾驶的需求，系统对道路情况进行数据采集，检测道路上的行人、机动车、非机动车，在其距离过近时提醒驾驶员以避免发生事故。

（4）路怒检测：针对目前部分驾驶员在驾驶过程中存在爆发路怒症的问题，当系统检测到驾驶员在驾驶过程中发生的愤怒的表情时，及时提醒驾驶员保持冷静。

（5）娱乐放松：为缓解车辆驾驶员在长时间驾驶后的疲劳感，本系统可以在驾驶员需要时通过语音交互播放音乐或进行其它娱乐活动。

（6）便捷化人机交互：本系统采用便捷的人机交互界面，操作简单，界面清晰，能够为司机提供良好的使用体验，降低因本系统自身使用的复杂性而产生的安全隐患。

2.2 设计原则

机动车作为现在大部分人的出行工具，驾驶员的驾驶行为受到社会的广泛关注，同时针对驾驶员在驾驶时经常出现的不安全驾驶行为以及兼顾驾驶员的驾驶体验，本系统的设计关注一下几点原则与理念：

（1）非侵入性原则：所谓非入侵性，是指系统在使用时，对使用者在物理上的约束性，是否有碍于使用者的正常活动，目前的一些疲劳驾驶监测系统将相关的监测装置安装于司机的身上，使用接触式的监测装置。这会极大的影响驾驶员的正常驾驶，降低驾驶员的驾驶体验，虽然准确度较高，但成本高，操作复杂。本系统在检测的同时兼顾驾驶员驾驶体验，设计出基于车载摄像头与车内传感器的非入侵性系统。

（2）准确性原则：考虑到本系统的应用场景为机动车车内，并对驾驶员的一些行为作出判断与检测，故系统的准确度是本系统应用时十分重要的性能之一。降低疲劳，酒驾的误判率能够提高驾驶员驾驶的专注度，准确，实时的对危险驾驶行为进行预警能够切实降低交通安全事故的发生率。基于以上考虑，本系统在实际应用中应有较高的准确率。

（3）人性化原则：驾驶员作为本系统的受众，不仅仅是检测的对象，也是本系统功能的使用者，在设计系统时，充分考虑了驾驶员的使用体验，在检测的基础上设计了智能语音交互与相应的娱乐功能，既能保障安全，又能提高驾驶员的驾驶体验。

2.3 功能与指标

（1） 疲劳检测

利用计算机视觉，基于YOLO4的深度学习架构，对人脸的特征点进行提取，并确定眼睛和脸部轮廓的位置，通过计算EAR值来判断司机是否疲劳。

（2） 酒驾检测

通过相应的酒精传感外设获取司机口中的酒精浓度，并设定相应的阈值判断是否酒驾。

（3）抽烟检测

基于OPENVINO工具包中的openpose模型进行人体特征点检测，同时进行相应的物体识别判断司机是否在吸烟。

（4）拨打电话监测

利用openpose进行相应的人体特征点提取，通过判断手的摆放位置进行驾驶员是否拨打电话的检测。

（5）路况检测

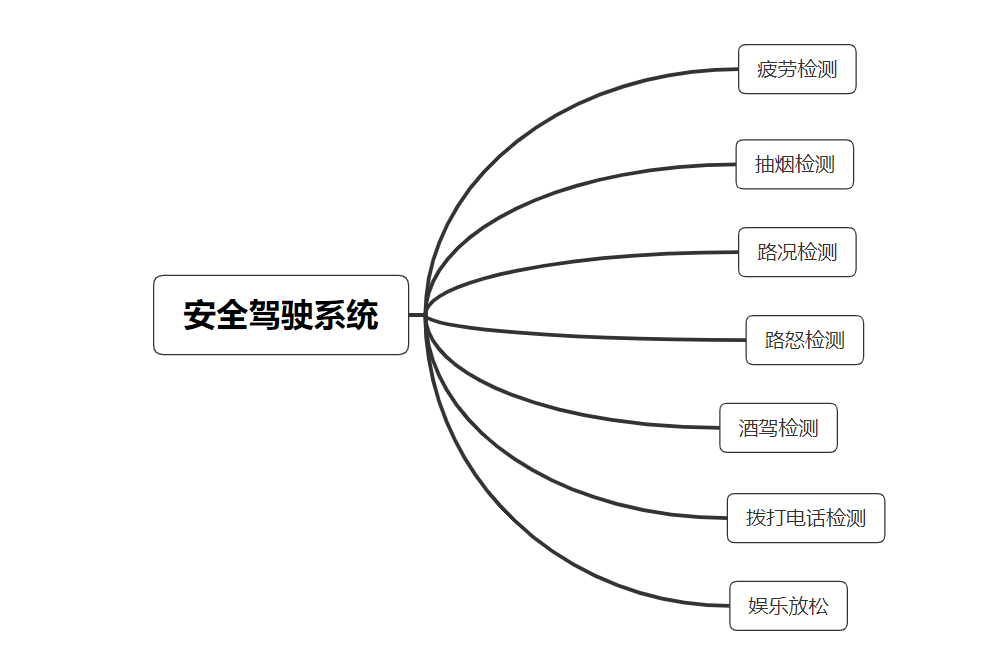
利用OPENVINO工具包中的Mobilenetv2模型作为预训练模型，识别道路上的行人、机动车和非机动车，在距离超过安全阈值时及时提醒，降低因驾驶员注意力不集中造成的安全隐患。

（6）路怒检测

利用OPENVINO工具包中的Mobilenetv2模型作为预训练模型，识别驾驶员在驾驶过程中的驾驶精神状态，当检测的驾驶员存在愤怒的情况时，及时提醒从而使驾驶员保持冷静。

（7）娱乐放松

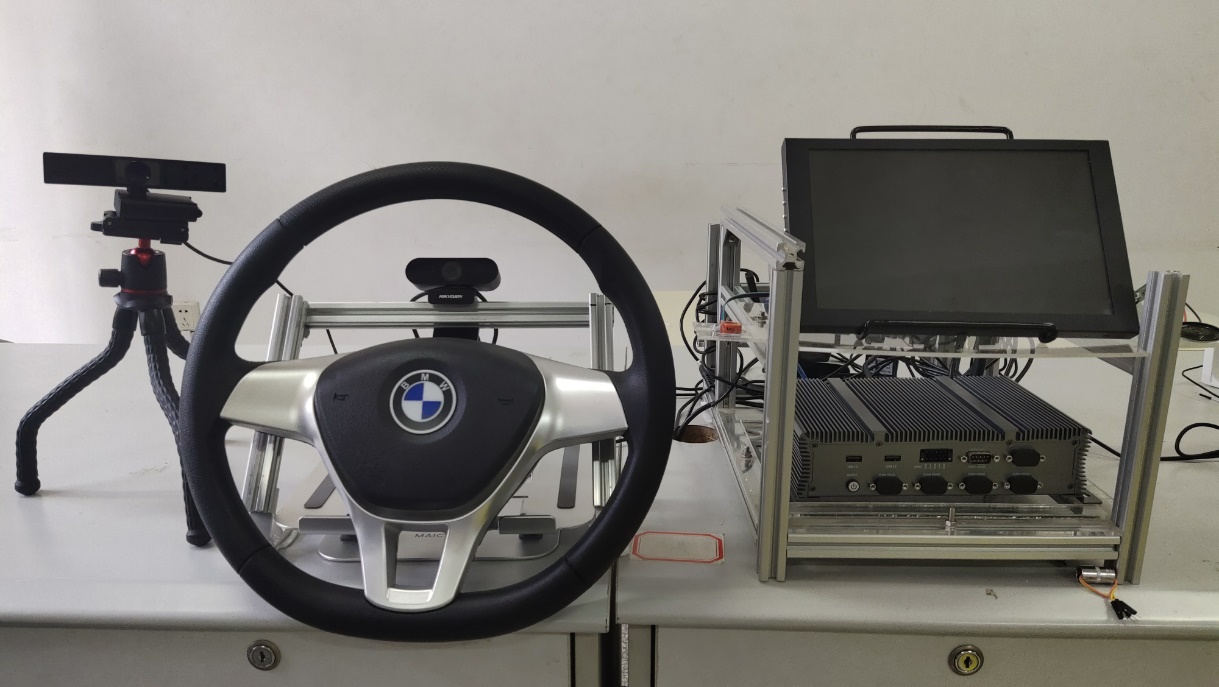
系统配备简介的UI界面与语音交互功能，使得司机能够通过触摸屏或语音的方式进行音乐的播放，提高司机的驾驶体验。



**图2-1 系统功能**

2.4 系统组成

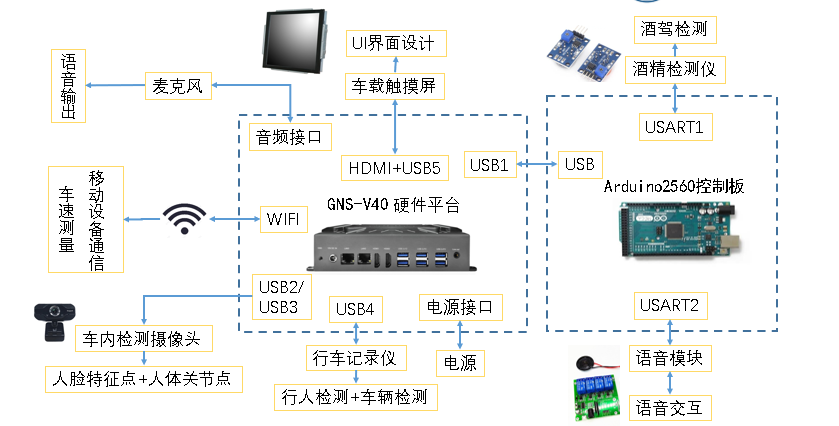
系统以GNS-V40硬件平台为核心，主要包括USB监控摄像头、麦克风、扬声器、触摸显示屏、ld3322模块、酒精传感器等外设组成，系统实物如图2-2所示。



**图2-2 系统实物图**

2.5 硬件框图

系统基于GNS-V40硬件平台，并通过相应的下位机实现与酒精检测仪，语音模块等相关外设的通信。三个车载摄像头（一个拍摄司机全身，一个拍摄司机面部，一个拍摄车外）通过USB接口与硬件平台连接，实现视频传输，同时配备的车载触摸显示屏通过HDMI接口与主机连接，实现UI界面的显示，为驾驶员提供便捷的操作。除此之外，系统通过音响与麦克风音频接口与硬件平台连接，为语音交互的设计提供可能，实现了智能化，人性化的交互方式。本系统还可连接WIFI实现数据的无线传输，并搭配相应的车载娱乐功能，下图为本系统的总体设计框图：

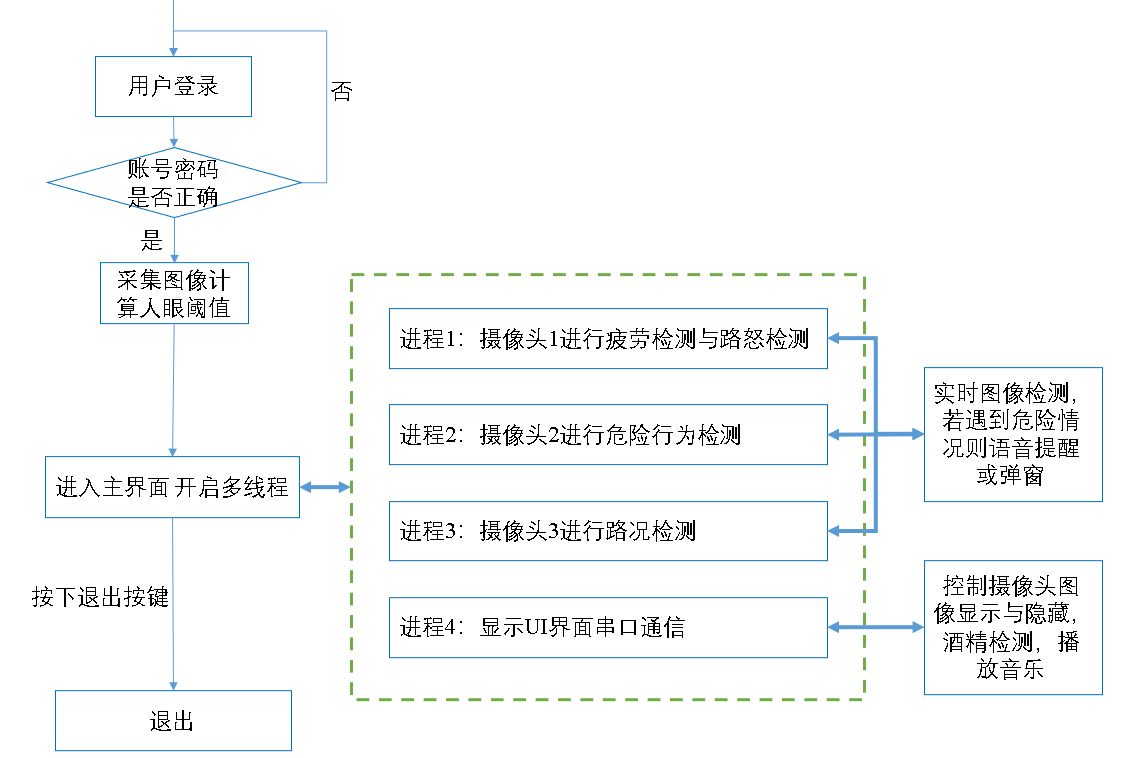


**图2-3 系统硬件框图**

系统充分利用了GNS-V40的各种端口资源，并结合一定的外设完成了系统的嵌入式设计。同时处理多个摄像头的视频数据也将充分利用主机的CPU性能，软件处理上使用多线程操作，将每个视频流交由独立的线程处理，极大的减小了CPU的等待时间，提高了视频处理的速率，优化系统性能。

2.6 软件流程

目前市场上主流的车载系统兼顾驾驶员的操作便捷与功能多样性，有提高整车控制集成度，提高信号传递精度，维修方便的特点。本系统切合目前车载系统的主要特点，整体操作易于上手，系统的软件流程图如下所示。



**图2-4 系统的软件流程图**

当系统启动后，驾驶员输入个人信息，登录后开始进行脸部图像的初始采集，采集结束后进入主程序，此时系统一共开启四个进程：

（1）进程一：开启摄像头采集驾驶员的脸部信息判断驾驶员是否疲劳或路怒；

（2）进程二：在第三个进程中开启摄像头采集驾驶员的行为信息，用于判断驾驶员是否出现例如抽烟，打电话等不规范的驾驶行为；

（3）进程三：开启检测车外路况场景的摄像头，用于判断车辆前方近距离处是否有行人，非机动车及机动车。

（4）进程四：实时检测触摸屏工作状态与串口信息的实时采集；

在前三个进程中若发现任何的危险行为，则会通过UI界面或语音进行相应的预警提示；当检测到路怒时，系统还会自动播放舒缓的音乐。同时，对于疲劳检测和路怒检测，驾驶员可以通过界面上的按钮实时切换，三个摄像头的视频信息也均可通过按钮进行调出与关闭。除上述的主要软件功能与流程外，系统还能为驾驶员提供当前时间与天气等信息，更加满足驾驶员的驾驶体验。

1. 系统原理及方案

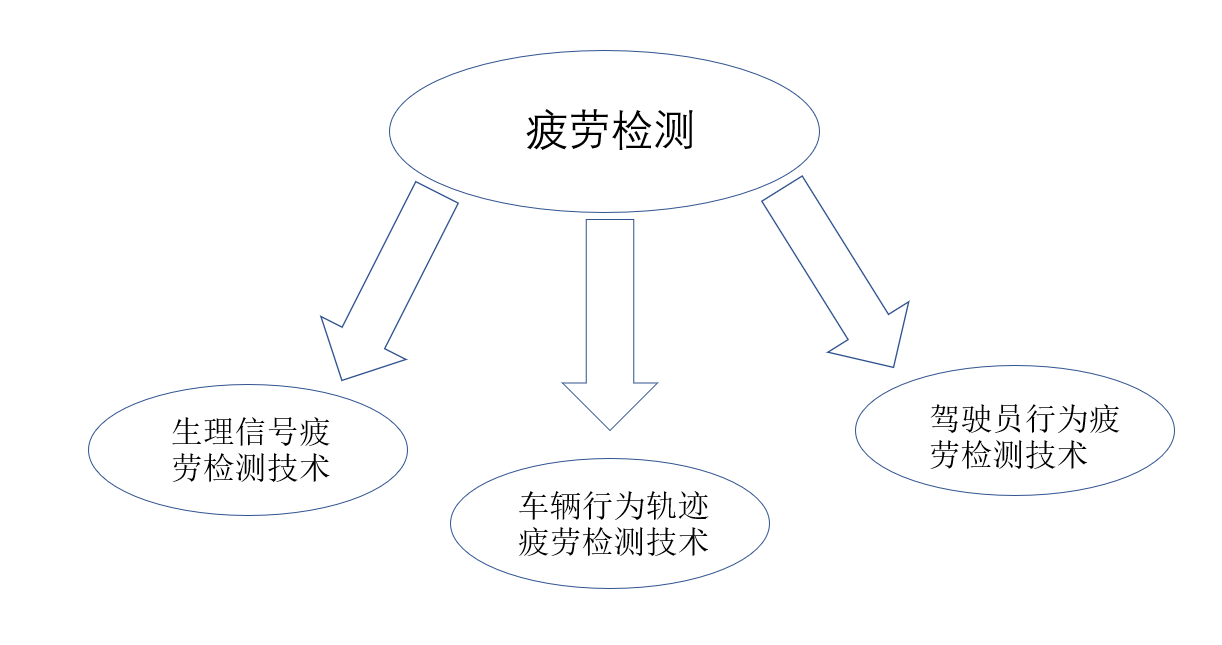
在经过了市场调研后，为满足驾驶员的日常需求，本团队提出了系统的功能，分别为疲劳检测、危险行为检测、路怒检测、路况检测等，接下来对系统各项功能的原理及实现方案进行了具体的阐述。

3.1 疲劳检测

随着计算机视觉技术的发展，人脸特征点检测受到许多研究人员的关注，近年来取得了一定的发展。事实上，驾驶行为识别、异常行为检测技术在智能驾驶、安全驾驶等领域具有广阔的发展空间，对广大机动车驾驶员能够起到有效的监督作用。系统基于Dlib库，在驾驶员驾驶过程中进行疲劳检测，发现驾驶员产生疲劳状况后进行语音报警，及时提醒驾驶员，为驾驶员驾驶安全保驾护航。

3.1.1研究综述

经过查阅、研究、总结国内外研究人员在驾驶员疲劳检测上的研究可以发现，在疲劳检测技术方面，主要可分为基于驾驶员生理信号的疲劳检测方法、基于车辆行为轨迹的疲劳检测方法和基于驾驶员行为（计算机视觉）的疲劳检测方法。



**图3-1 疲劳检测常用方法**

（1）基于驾驶员生理信号的疲劳检测技术

关于疲劳驾驶的研究者早期致力于从疲劳机理与生理特征进行研究。从生理学的角度，当驾驶员发生疲劳时，人体的各项生理指标会出现不同程度上的变化。主要包括脑电信号检测、心电信号检测、肌肉信号检测三种。

其中，脑电信号检测中，最初凯克顿等人发现从清醒状态过渡到疲劳状态脑电图会发生明显的变化，这也为通过脑电信号检测疲劳提供了理论支撑。之后，科研人员在实验中，采用了一款脑电仪的仪器设备，利用该设备对驾驶员的脑电信号等数据进行监测，再使用功率谱分析脑电信号在不同频段波的变化规律，实验结果[13]表明：跟踪检测的波能量变化是可以作为疲劳检测的评价指标的；而心电信号检测则利用心电图机可以测出心脏每一周期跳动时的波形，当驾驶员处于疲劳时，可通过分析心率变异性（HRV）的变化来评估驾驶员的疲劳程度；肌肉电信号检测则为驾驶员在长时间驾车行驶后，由于神经处于高度紧张状态，所以经常伴有肌肉僵硬或者腰酸背痛之类的现象，而在疲劳时肌肉收缩力减小，肌肉呈放松状态，故通过设备观测驾驶员肌肉神经的活动情况来判断是否处于疲劳状态。

驾驶员的生理信号实际上最能反映其自身的真实状态，该检测方法抗干扰性比较强、准确性高、实时性较好，但缺点在于采集信号都需要接触驾驶员的身体肌肤，驾驶员对该种方案的主观感受是消极的，会认为佩戴过程繁琐、接触采集设备会产生不适感、长时间接触电极对身体造成一定的伤害等问题[14]，对研究人员来说研究成本较高，其中最重要的一点是接触式的检测方法容易干扰到驾驶员的正常驾驶，如果因为采集设备的干扰而造成安全事故，就会得不偿失。

（2）基于车辆行为轨迹的疲劳检测技术

当驾驶员在驾车过程中察觉到疲劳时，周围的环境在其眼里变得模糊，肌肉的松弛会使得对车辆的控制能力减弱，因此会致使车辆行驶的一部分数据与正常行驶状态的数据相比，会出现有较大的偏差，所以研究者可以将车辆行为轨迹作为检测目标来识别驾驶员的疲劳状态。目前基于车辆行为轨迹的疲劳检测方法主要是利用摄像头和传感器来监测车辆在行驶时的一些行为轨迹数据的变化，如方向盘的偏角、行驶速度以及车道偏离的角度等。

在相关研究人员的研究中，通过采集了驾驶员在不同疲劳状态下驾驶时的方向盘角度偏向，根据这些角度偏向数据绘制了不同的波形图，其中绘制的一幅波形图是记录了驾驶员从清醒到疲劳再到非常疲劳的一次行车过程，从图中可以明显的观察出体现在方向盘上的两个特征：修正的频数降低以及大幅度修正频数增加，这就表明驾驶员的疲劳特征确实能够反映在方向盘的转角变动上，并且结合 Fisher线性判别算法加判别准则，形成了实时检测疲劳驾驶的模型，并根据模型进行驾驶员疲劳检测[15]。

和生理信号的检测方法相比，此方法不需要和驾驶员接触，因此也不会扰乱驾驶员的正常驾驶，但是由于不同的驾驶员在年龄、驾驶方式以及驾驶经验等方面情况不同，

驾驶行为也会有很大的差异，从而使检测结果出现偏差，并且这种方法也很容易被天气、车辆特性以及行驶道路的状况等外界环境影响，鲁棒性不强。最重要的一个缺点是车行为轨迹出现偏差时，有可能已经处于危险状态，该方法并不能够达到及时提醒驾驶员的效果，并且检测的准确性一般，成本也较高，一般可用来做疲劳检测的辅助指标。

（3）基于驾驶员行为特征的疲劳检测技术

驾驶员从清醒状态进入疲劳状态时，其面部特征和头部的行为特征会发生明显的变化，因此，通过摄像头采集驾驶员面部和头部特征数据，通过计算机视觉分析的技术对其分析是一种合理的方案。该检测方法按照身体部位可分为三类：眼部的特征检测、嘴部的特征检测以及头部的特征检测。下面具体阐述三部分身体部位的特征检测。

其中眼部的特征检测通过计算机视觉分析对眼部特征进行提取，当驾驶员困倦时且没有完全失去意识之前，一般会通过快速的眨眼让自己保持清醒，导致眨眼次数增多。随着疲劳程度加深，驾驶员意识逐渐模糊，眨眼次数会相应的减少，在进入睡眠状态时，眨眼次数变为 0。驾驶员在驾驶过程中，80%～90%的信息都是通过眼睛获取的，也就是说视觉在驾驶员行车过程中为主要信息输入来源。驾驶员在单位时间内闭眼的时间严重超出其正常闭眼的时间，不论其是否处于疲劳状态，此类行为都有可能造成严重的交通事故，类似此行为的都可以作为眼部特征，用特征的变化作为评判驾驶员是否在疲劳驾驶的依据。

人在疲劳时，会出现打哈欠现象。嘴巴有多种状态，每种状态下的张开程度是不同的，所以可以通过嘴巴张开程度的大小以及张开的持续时间将打哈欠与其他嘴部状态区分开，进而通过计算打哈欠的频率来对是否属于疲劳驾驶进行判断。但是打哈欠这个特征是因人而异，不是处于疲劳状态就一定伴有打哈欠的行为，而且嘴部疲劳的特征检测实时性一般，不能作为判定疲劳的唯一参数，可以结合眼部的特征检测进行综合评判来判断疲劳。

头部姿态可以表示人脸的朝向以及人眼的注视方向，驾驶员在清醒状态下驾驶时，眼睛始终平视正前方，所以头部基本上是维持在一定的角度范围内。当查看周围行驶路况时，可能出现头部左右扭动，时间不超过 1 秒且扭动角度小于 45 度，或者是抬头通过后视镜查看行驶的路面情况。而在疲劳时，大脑对身体的控制能力减弱，会出现频繁的点头运动，不会出现大幅度的转头、摇头等动作。因此，利用疲劳产生的点头运动的频率研究疲劳驾驶也是合理的[16]。

该检测方法同样不与驾驶员进行接触，具有较强的抗干扰性和较高的准确性，实时性也很好，成本较低。同时，随着人工智能技术的不断普及，计算机视觉方面的检测也正成为目前主流的检测方法。考虑到GNS-V40硬件平台强大的视觉处理能力，本文选择基于驾驶员行为特征的检测技术进行疲劳检测，通过提取人脸特征点设计疲劳检测算法，力求在驾驶环境中实现较高的准确率。

3.1.2 疲劳检测算法分析

本系统通过处理人脸特征点来判断疲劳，人脸人脸特征点提取（Face Landmark Extraction），是通过将图片中已检测到的人脸提取出来，从而获得人脸特征点。人脸特征点通常对应人脸相对应的各个五官，比如眼、嘴、鼻等。



**图3-2 人脸68特征点图**

Dlib是一个现代的 C++工具包，包含机器学习算法和工具。它广泛应用于工业界和学术界，包括机器人，嵌入式设备，移动电话和大型高性能计算环境。Dlib人脸数据把人脸分成了68个数据点，每个人脸部位都有不同的标签，我们通过访问68个点就可以来获取人脸的部分数据[17]。下面分别阐述对眼部、嘴部、头部的处理。

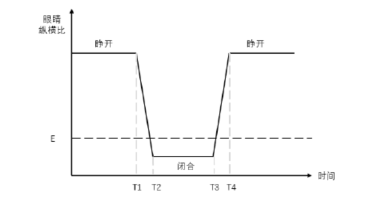
（1）眼部特征分析

眼部的特征主要反映在眼部开合度，若仅利用眼睛的高度判断眼睛的开合度，实际处理过程中可能会受到近大远小成像原理的影响，因此本团队采用计算眼睛纵横比（eye aspect ration，EAR）判断眼睛睁开大小的程度，能够有效的消除图像成像原理的影响，并且 EAR 的值对头部的姿势并不敏感。

左眼和右眼的EAR需要分别计算，公式如下所示：

一般情况下，人在眨眼时，左右眼会同步进行这一动作。因此，可以利用两只眼睛的 EAR 平均值作为最终的 EAR 值，如下所示：

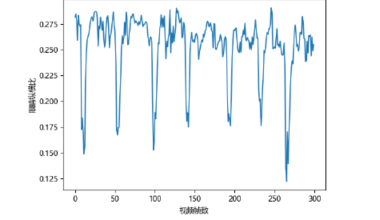
眼睛处于睁开状态时，眼睛的宽度、高度保持不变，故 EAR 值也保持恒定；眼睛处于闭合状态时，眼睛的宽度不变，高度最小，甚至为 0，因此 EAR 值也基本接近于零。人在正常情况下需要通过眨眼来保护眼睛，避免被灰尘损伤，当眼睛发生眨眼动作时，EAR 也会随之变化，我们将眼睛从完全睁开到完全闭合再到完全睁开连续动作认为是一次完整的眨眼过程，下图所示为一次完整眨眼过程的 EAR变化示意图。



**图3-3 单次眨眼时EAR变化曲线图**

眼睛在闭合到一定程度时，对外部环境的感知能力就变弱，因此需要设定一个阈值 E 来区分眼部两种不同状态。当眼睛的 EAR 大于阈值 E 时，认为眼睛处于睁开状态，否则认为其处于闭合状态。T1 时刻前的一段横线表示眼睛处于睁开状态，此刻为开始眨眼的起始时刻；T1 到 T2 时间段内，眼睛迅速从睁开状态变成闭合状态，EAR 值也会迅速下降，直到眼睛完全闭合；T2 到 T3 时间段内，眼睛处于闭合状态，但不会持续太长时间，并且此时的 EAR 处于整个眨眼过程中的最低值；T3 到 T4 时间段内，眼睛从闭合状态恢复到睁开状态，EAR 随之增大；T4 时刻之后，眼睛处于睁开状态。至此，从 T1 到 T4 完成了一次完整的眨眼动作[18]。

在确定眼部特征提取方案后，阈值E的选取成为确定眼部状态的准确性的一个重要因素，为了将EAR阈值设定的更加合理和真实，本文进行了EAR阈值的设定实验，通过收集驾驶员在清醒状态下的驾驶视频，并对采集的视频进行人脸检测，定位人脸关键点，并通过计算眼部特征值，获得每一帧的EAR值，以帧数为横坐标，EAR值为纵坐标，来判断人眼状态变化，如下图所示



**图3-4 清醒时EAR变化曲线图**

将其波谷的最低点作为完全闭眼的阈值，根据PERCLOS-P80 准则，将EAR阈值设定为完全闭眼的阈值的80%。

（2）嘴部特征提取

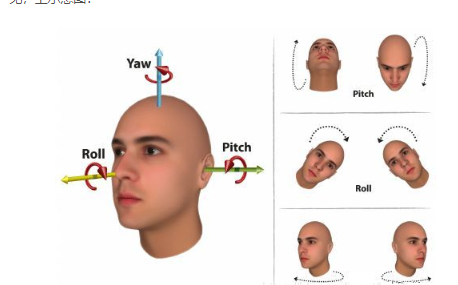
和眼部状态识别类似，本文采用嘴巴纵横比（Mouth Aspect Ratio, MAR）来区分嘴部的状态，即计算嘴巴张合的程度，同时为了避免因嘴唇厚度而造成的误差，本文采用嘴部内轮廓的关键点来计算MAR，公式如下图所示

嘴巴在张开时，上下嘴唇的间距变大，嘴角间的宽度变小，由公式（3-4）可知，嘴巴张开的程度越大 MAR 的值就越大；嘴巴紧闭时，此时上下嘴唇的间距最小，MAR 的值也是最小，理论上可以达到 0；说话时的 MAR 值大于闭合时的 MAR值，一般小于打哈欠时的 MAR 值。因此可以找一个阈值 M 将打哈欠与嘴巴正常闭合时的状态和说话时的状态区分开，从而完成对嘴部状态的识别。需要注意的是，说话时的 MAR 值不一定都小于阈值 M 的，即说话时的 MAR 值在某一时刻大于阈值 M 的情况是存在的，由于说话时嘴巴都是一张一合，即使在某一时刻大于阈值 M，其持续时间也会非常短暂，因此本文采用双阈值法检测哈欠：结合嘴巴张开的程度与持续时间进行哈欠检测，即当 MAR 大于阈值 M 并且持续一定的时间Mt时，认为发生了一次打哈欠动作。

在MAR阈值的选取过程中，选取开源数据集YawDD中打哈欠的视频，利用嘴部张合程度计算方法计算MAR值。选取打哈欠过程中的加权平均值得到打哈欠的阈值。

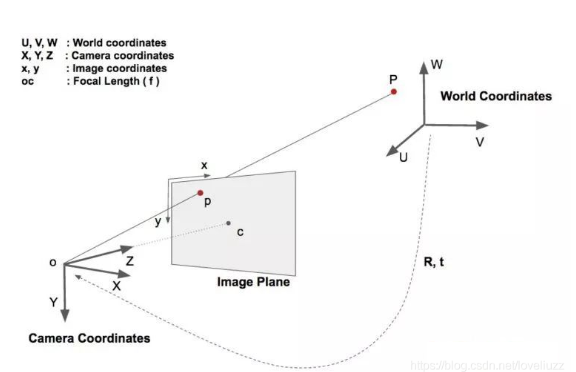
（3）头部特征提取

通过一幅面部图像来获得头部的姿态角. 在3D 空间中，表示物体的旋转可以由三个欧拉角(Euler Angle)来表示：分别计算pitch(围绕X轴旋转)，yaw(围绕Y轴旋转)和roll(围绕Z轴旋转) ，分别学名俯仰角、偏航角和滚转角，通俗讲就是抬头、摇头和转头。



**图3-5 头部姿态示意图**

在头部姿态估计中，采用Head Pose Estimation 算法，具体的算法步骤包括（1）2D人脸关键点检测；（2）3D人脸模型匹配；（3）求解3D点和对应2D点的转换关系；（4）根据旋转矩阵求解欧拉角。在2D到3D转换过程中，采用坐标系转换的方法



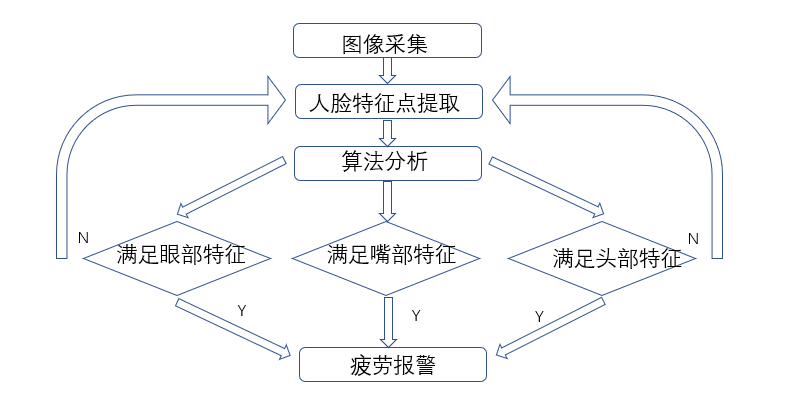
**图3-6 坐标变换示意图**

通过获取人脸特征点坐标位置以及相机参数从而求解旋转矩阵，判断欧拉角。其中由中心坐标系到像素坐标系的关系如下所示

驾驶员在驾驶过程处于疲劳时，不仅在眼部和嘴部上有异于常态的行为动作发生，在头部姿态上也有一定的变化。在清醒状态驾驶的情况下，驾驶员需要随时对外界保持一定的注意力，在车流量较大的道路上，驾驶员会灵活的转动头部以便从正前方、左右倒车镜以及后视镜中查看当前路况，在车流量较少的高速公路上直线行驶时，驾驶员通常会保持恒定的头部状态，而驾驶员在感觉疲劳时，注意力会下降，头部活动范围较小，并且有时会伴随瞌睡点头的现象的发生，因此本文将瞌睡点头的频率作为头部的疲劳特征，即关注 pitch 姿态角的变化，当 pitch 超出正常活动范围则认为驾驶员发生了一次点头动作。

在完成识别眼、嘴、头三个部位的状态，还需要选择判定疲劳状态的参数，通过实时计算这些疲劳参数，并结合各参数设定的阈值，就能完成一次对驾驶员是否处于疲劳状态的判断。

本系统基于PERCLOS进行疲劳判定，PERCLOS 是当前公认的最有效的、应用最广泛的评定疲劳状态方法，结合PERCLOS的阈值选取方法，即P80，当上眼睑覆盖瞳孔80%以上的面积时，认为此时眼睛是闭合的，计算单位时间内闭眼时长所占的比重；小组提出了针对疲劳的判断方法计算30秒时间段内的 PERCLOS、眨眼频率以及持续闭眼时间，若 PERCLOS 超出 40%并且持续闭眼时间长于3秒，则认为驾驶员已经进入了疲劳阶段。通过嘴部判定疲劳的参数阈值选取：打哈欠是一个长时间持续张大嘴巴的过程，并且打哈欠的间隔时间较长。本文将单位时间设定为 30 秒，选取的哈欠参数阈值为2，即在疲劳检测的过程中对单位时间内打哈欠的次数进行计数，若驾驶员在此段时间内打哈欠的次数超过2个，则认为其进入疲劳阶段。通过头部判定疲劳参数的阈值选取：瞌睡点头作为头部的疲劳特征，疲劳检测会实时计算瞌睡点头的频率，本团队将瞌睡点头的频率设定为5，即30秒内点头次数大于5则会被认定为疲劳。疲劳检测的总流程如下图所示



**图3-7 疲劳检测流程图**

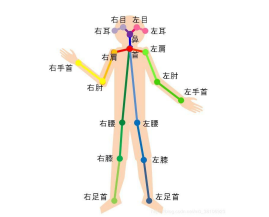
本系统通过采集驾驶员的驾驶图像，对驾驶员的人脸特征点进行提取，并结合上述算法计算驾驶员的眼部特征、嘴部特征、头部特征，综合判断驾驶的疲劳状态，当检测到驾驶员出现驾驶疲劳时及时进行提醒。本系统综合驾驶员的眼部特征、嘴部特征、头部特征进行判断，能够更加快速准确且有效的判断驾驶员的疲劳状态。

3.2 危险行为检测

随着计算机视觉技术的发展，人体姿态估计受到许多研究人员的关注，近年来取得了一定的发展。事实上，日常行为识别、异常行为检测技术在智能驾驶、智慧安防等领域具有广阔的发展空间，对驾驶员人群能够起到有效的监督作用。系统基OpenPose库，在驾驶员驾驶环境中进行危险行为检测，发现接打电话、抽烟等危险行为后进行及时报警提醒驾驶员，终止危险行为，为驾驶员驾驶安全保驾护航。

3.2.1 Openpose原理

人体姿态估计（Human Pose Estimation），是通过将图片中已检测到的人体关键点正确的联系起来，从而估计人体姿态。人体关键点通常对应人体上有一定自由度的关节，比如颈、 肩、肘、腕、腰、膝、踝等。 OpenPose人体姿态识别项目是美国卡耐基梅隆大学（CMU）基于卷积神经网络和监督学习并以Caffe为框架开发的开源库，可以实现人体动作、面部表情、手指运动等姿态估计，适用于单人和多人，具有极好的鲁棒性，是世界上首个基于深度学习的实时多人二维姿态估计应用，基于它的实例如雨后春笋般涌现[19]。

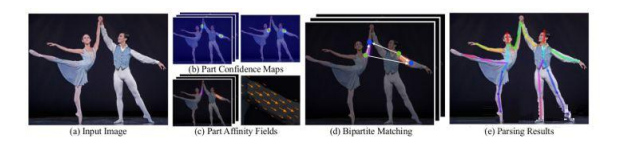


**图3-8 人体特征点图**

OpenPose的实现原理包括以下三方面：

（1）输入一幅图像，经过卷积网络提取特征，得到一组特征图，然后分成两个岔路，分别 使用CNN网络提取关节点置信图（PCM）和关节亲和场（PAF）；

（2）得到这两个信息后，使用图论中的偶匹配求出关节联系，将同一个人的关节点连接起 来，由于PAF自身的矢量性，使得生成的偶匹配很正确，最终合并为一个人的整体骨架；

（3）最后基于PAF求多人关节点解析，把多人关节点解析问题转换成图问题，利用匈牙利算法求解。 

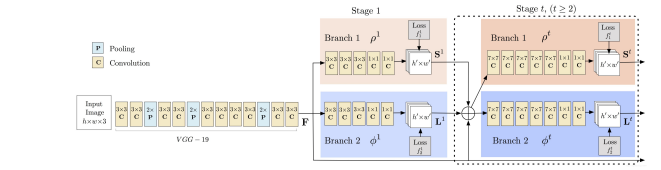
**图3-9 OpenPose实现原理**

实现的神经网络包括以下三个阶段：

阶段一：VGGNet的前10层用于为输入图像创建特征映射。

阶段二：使用2分支多阶段CNN，其中第一分支预测身体部位位置（例如肘部，膝部等） 的一组2D置信度图（S）。第二分支预测一组部分亲和度的2D矢量场（L），其编码部分之间 的关联度。

阶段三：通过贪心推理解析置信度和亲和力图，对图像中的所有人生成2D关键点。



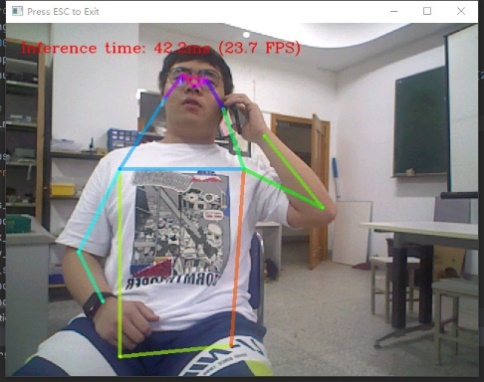
**图3-10 OpenPose网络结构**

3.2.2基于人体关键点的危险行为检测算法

系统以计算驾驶员手部特征点到耳部特征点的距离与鼻特征点的距离判断驾驶员危险行为事件的发生，利用 OpenPose 逐帧处理来自摄像头的视频流，提取出人体的鼻、耳、眼、颈、肩、肘、腕关键点[20]，通过设计的算法分析关键点生成姿态特征，若满足检测特征之一，则判断驾驶员存在危险驾驶行为。

（1）判断打电话行为

系统通过提取与接打电话相关的头部、肘部、腕部三个关键的特征点，利用这些点计算驾驶员手部到耳部的距离来判断抽烟行为的发生。

定义两个向量D1与D2分别描述左手到左耳的距离以及右手到右耳的距离，根据驾驶员接打电话的运动特性，通过判断D1与D2来判断驾驶员是否存在接打电话行为。

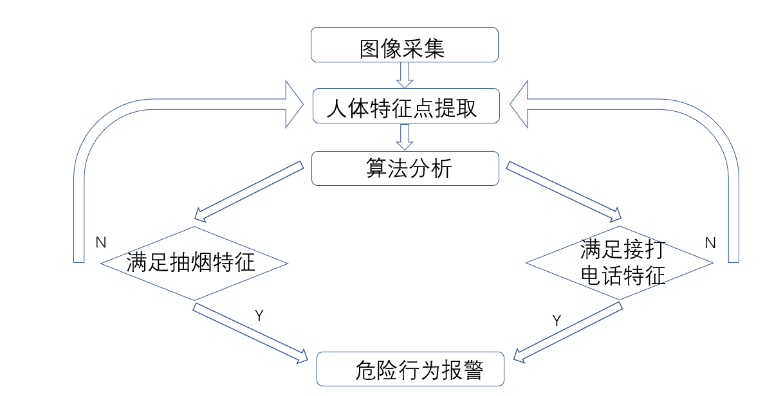
**图3-11 OpenPose实际效果图**

（2）判断抽烟行为

系统通过提取与抽烟相关的耳部、肘部、腕部三个关键的特征点，利用这些点计算驾驶员手部到耳部的距离来判断抽烟行为的发生。

定义两个向量D1与D2分别描述左手到鼻子的距离以及右手到鼻子的距离，根据驾驶员抽烟的运动特性，通过判断D1与D2来判断驾驶员是否存在抽烟行为。

系统整体的流程如下图所示



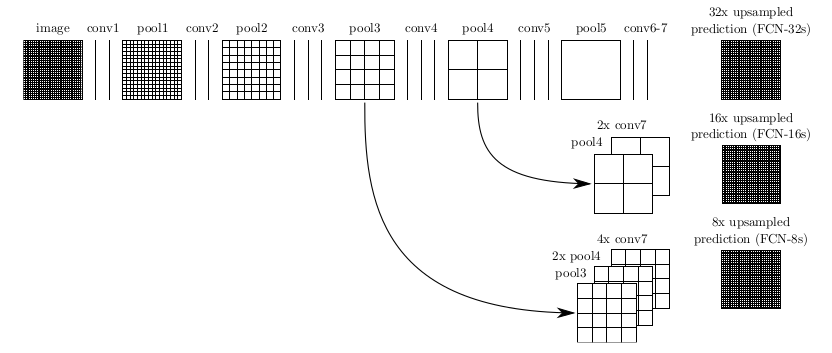
**图3-12 危险行为检测流程图**

本套算法在检测驾驶员的危险行为时具有较好的鲁棒性，可有效适应不同的车内环境，当驾驶员姿态特征满足抽烟或者接打电话特征时，判断驾驶员存在危险驾驶行为。此时通过语音提醒驾驶员存在危险驾驶行为。

3.3 情绪检测

目前驾驶员在驾驶过程中出现路怒的情况时有发生，系统基于计算机视觉，采用openvino工具包中的情绪识别模型，利用全卷积层神经网络实现了对驾驶员情绪的实时判断，模型采用的数据集为AffectNet数据集（该数据集中包含大量的人脸情绪图像，共计120GB，本模型即取用了五种情绪的人脸图像作为数据集，总计2500张），通过该模型能够判别驾驶员的正常，高兴，悲伤，惊讶，生气五种情绪。该模型使用全卷积层神经网络FCN（Fully Convolution Network），FCN作为语义分割的先河，能够实现像素级别的分类，为之后使用CNN 作为基础的图像语义风格模型提供重要的基础。FCN相对于普通CNN网络，将其中的全连接层全部换成了卷积层，

在FCN中利用了上采样的过程，使得每一次采样后数据数量增多。常用的有三种上采样的方法，使用其中的反卷积的方法可以使得图像实现end to end.如下展示了如何实现上采样的过程



**图3-13 FCN处理流程**

上图分别描述了使用FCN-32s,FCN-16s,FCN-8s的过程，image表示原图像，conv1,conv2..,conv5为卷积操作，pool1,pool2,..pool5为池化操作。其中第一行对应FCN-32s,第二行对应FCN-16s,第三行对应FCN-8s. 全卷积网络的核心贡献在于使用卷积层通过学习让图片实现了end to end的分类特性，正是由于这一特点，我门采用基于FCN的emotions-recognition-retail-0003模型来进行对驾驶员情绪的判别，通过车载摄像头进行图像采集，并将每一帧的图像输入到模型中，利用openvino工具包进行加速，提高对司机情绪的判别速率，情绪识别模型的相关说明如下：

**表3-1 情绪识别模型相关标准**

|  |  |
| --- | --- |
| 标准 | 相关参数 |
| 输入的人脸方位 | 前方 |
| 面内角度（rotation in-plane） | ±15° |
| 面外角度（rotation out-of-plane） | Yaw:±15°/Pitch:±15° |
| 最小人脸宽度 | 64像素 |
| 源框架 | Caffe |

该模型测量的准确度为70.20%,其中模型的输入为三通道，64\*64的图像类型，图像采用BGR模式，将图片或视频流输入模型后得到的输出为1\*5\*1\*1的矩阵类型，其中标签“0”表示正常，标签“1”表示高兴，标签“2”表示悲伤，标签“3”表示惊讶，标签“4”表示愤怒[21]。

3.4 路况检测

随着近年来自动驾驶的发展，道路检测也越来越受到研究人员的关注，但是由于车外环境干扰太多，提高道路检测的准确度一直以来都是困扰研究人员的一大难题。本系统采用OPENVINO工具包的Mobilenetv2模型对其道路上的行人、机动车和非机动车检测，同时利用基于单目摄像头的测距原理车辆检测目标与当前摄像头的距离，在其超过阈值时报警。

3.4.1 Mobilenetv2模型原理

Mobilenet是谷歌团队2017年提出的，专注于移动端或者嵌入式设备中的轻量级CNN网络，在准确率只有极小幅降低的情况下，大量减少参数与运算量。MobileNetV2是对MobileNetV1的改进，是一种轻量级的神经网络。MobileNetV2保留了V1版本的深度可分离卷积，增加了线性瓶颈（Linear Bottleneck）和倒残差（Inverted Residual）。

MobileNetV2的模型如下表所示，其中t为瓶颈层内部升维的倍数，c为特征的维数，n为该瓶颈层重复的次数，s为瓶颈层第一个conv的步幅。

**表3-2 MobileNetV2模型**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Operator | t | c | n | s |
|  | conv2d | - | 32 | 1 | 2 |
|  | bottleneck | 1 | 16 | 1 | 1 |
|  | bottleneck | 6 | 24 | 2 | 2 |
|  | bottleneck | 6 | 32 | 3 | 2 |
|  | bottleneck | 6 | 64 | 4 | 2 |
|  | bottleneck | 6 | 96 | 3 | 1 |
|  | bottleneck | 6 | 160 | 3 | 2 |
|  | bottleneck | 6 | 320 | 1 | 1 |
|  | conv2d 1\*1 | - | 1280 | 1 | 1 |
|  | avgpool 7\*7 | - | - | 1 | - |
| 1\*1\*1280 | conv2d 1\*1 | - | k | - |  |

除第一层外，整个网络中使用恒定的扩展率。在实验中，发现在 5 到 10 之间的扩展率会导致几乎相同的性能曲线，较小的网络在扩展率稍低的情况下效果更好，而较大的网络在扩展率较大的情况下性能稍好。

MobileNetV2主要使用 6 的扩展因子应用于输入张量的大小。 例如，对于采用 64 通道输入张量并产生具有 128 通道的张量的瓶颈层，则中间扩展层为 64×6 = 384 通道。

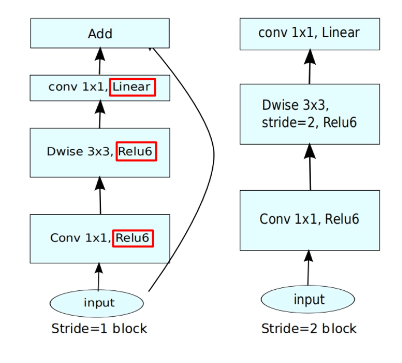
而线性瓶颈指的是，对于MobileNetV1的深度可分离卷积而言， 宽度乘数压缩后的M维空间后会通过一个非线性变换Relu，根据Relu的性质，输入特征若为负数，该通道的特征会被清零，本来特征已经经过压缩，这会进一步损失特征信息；若输入特征是正数，经过激活层输出特征是还原始的输入值，则相当于线性变换。瓶颈层的具体结构如下表所示。输入通过1的conv+Relu层将维度从k维增加到tk维，之后通过3×3conv+Relu可分离卷积对图像进行降采样（stride>1时），此时特征维度已经为tk维度，最后通过1\*1conv（无Relu）进行降维，维度从tk降低到k维。

**表3-3 MobileNetV2的输入与输出**

|  |  |  |
| --- | --- | --- |
| Input | Operator | Output |
| h\*ω\*k | 1\*1 conv2d,Relu6 | h\*ω\*(tk) |
| h\*ω\*tk | 3\*3 dwise s=s,Relu6 | \* |
| \* | Linear 1\*1 conv2d | \* |

残差块已经在ResNet中得到证明，有助于提高精度构建更深的网络，所以MobileNetV2也引入了类似的块。经典的残差块（residual block）的过程是：1x1(降维)–>3x3(卷积)–>1x1(升维)， 但深度卷积层（Depthwise convolution layer）提取特征限制于输入特征维度，若采用残差块，先经过1x1的逐点卷积（Pointwise convolution）操作先将输入特征图压缩，再经过深度卷积后，提取的特征会更少。所以MobileNetV2是先经过1x1的逐点卷积操作将特征图的通道进行扩张，丰富特征数量，进而提高精度。这一过程刚好和残差块的顺序颠倒：1x1(升维)–>3x3(dw conv+Relu)–>1x1(降维+线性变换)。

结合上述线性瓶颈和倒残差的理解，Mobilenetv2模型结构图如下所示，先通过1x1的卷积进行升维，再经过3x3的卷积，再经过1x1的降维。形成瓶颈结构，激活函数采用Relu函数。当stride=1时，才会使用elementwise 的sum将输入和输出特征连接（如下图左侧）；stride=2时，无short cut连接输入和输出特征（下图右侧）。



**图3-14** **Mobilenetv2模型结构图**

3.4.2 单目测距原理

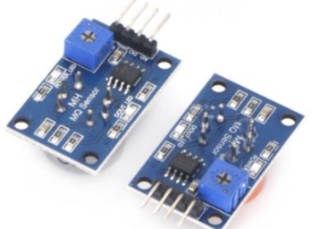
自动驾驶场景或者Adas场景中，当检测出前方车辆后通常需要进行距离估计，为车辆控制提供距离参考信息；而基于视觉的移动机器人也存在该需求。单目摄像头光学图像测距具有低成本和计算快的优点，主要有两种常用的测距方式：使用目标物体大小和摄像头焦距；使用摄像头高度和俯仰角。本系统采用使用目标物体大小和摄像头焦距测距。单纯的单目视觉测距，必须已知一确定长度。假设此时目标物体宽度为W，并与相机相距D。利用相机对物体拍照并测量物体像素宽度P,相机焦距公式为：

（3-6）

得到相机焦距后，在实际过程中，利用摄像头采集前方道路图像，并检测，通过矩形框将物体框定出来。结合矩形框信息，找到矩形框底边的两个相平面坐标，通过欧氏距离和公式（3-6）的相关信息计算得到目标与相机的距离，之后当估测出的距离达到一定的阈值时，便会进行相应的警告与提示。

3.5 酒精测量

酒驾现在仍然是大部分交通事故发生的主要原因，本系统除视觉部分之外，也通过搭配相应的酒精传感器来对司机的酒驾行为进行检测。在设计中，下位机通过数模转换模块实时检测酒精传感器AO口的输出值，通过判断当前空气中酒精浓度来判断是否酒驾。酒驾时每百毫升血液酒精含量大于20毫克，而醉驾则是血液中酒精含量大于80mg/100ml。根据此标准，当空气中酒精含量超过一定阈值时，下位机则通过串口与主机通信，并基于相应的弹窗警告和语音警告。



**图3-15 酒精模块实物图**

本系统酒精测量仪使用的芯片为AT89S52，测量范围为10-10000ppm,AO口的输出电压为0.1-3. 2V，空气中酒精浓度越高，其输出电压也越高。司机在上车后需要对酒精传感器进行吹气，依据本系统的测试与模拟，当传感器AO口的输出值大于300时即认为司机体内的酒精浓度过高，属于酒驾[22]。只有当司机吹气时的酒精含量达标才可继续驾驶汽车。

技术参数

（1）标准测试条件

·温度：20℃±2℃

·相对湿度：65%±5%RH

·标准测试电路：Vc:5.0V±0.1V

·预热时间：≥48h

（2）检测浓度：0.04mg~4mg/L酒精蒸气

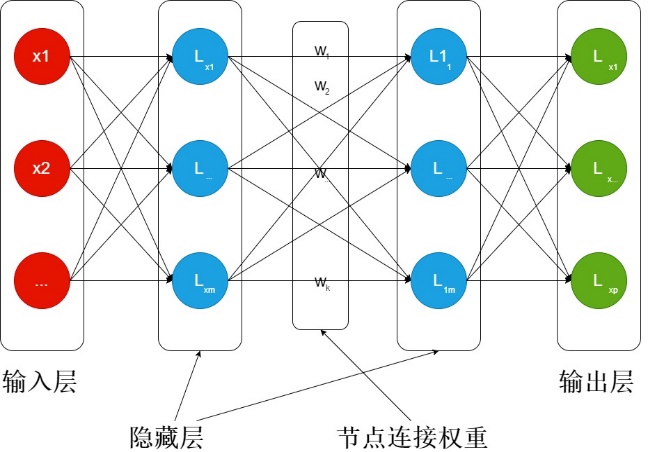
（3）灵敏度：Rs(in air)/Rs(0.4mg/L酒精)≥5

3.6 人机交互与多媒体娱乐

司机在驾驶中其注意力应主要集中于道路情况，为此系统结合了百度语音API与相应的语音模块实现与驾驶员的语音交互，并通过PYQT库设计了操作简洁的交互界面，以供驾驶员更加便捷的使用。并添加了音乐播放的功能，给驾驶员提供更好的驾驶体验。

3.6.1 基于深度神经网络的语音识别

人工智能的基础是深度学习，而深度学习又是机器学习的一个分支，是一种使用多重线性变换形成多个神经网络处理层复杂结构，进而对数据进行处理与高层抽象的算法[32]。基于深度神经网络的语音识别系统能够有效地利用持续增长的数据量，提升识别性能，激发更多的产业应用，打通语音数据的获取渠道，从而进一步优化模型。深度神经网络内部的神经网络可以分为三类：输入层、隐藏层和输出层，如下图3-16所示。



**图3-16 语言识别深度神经网络内部结构图**

在深度神经网络（DNN）中层与层是全连接的，其局部模型是一个线性关系：

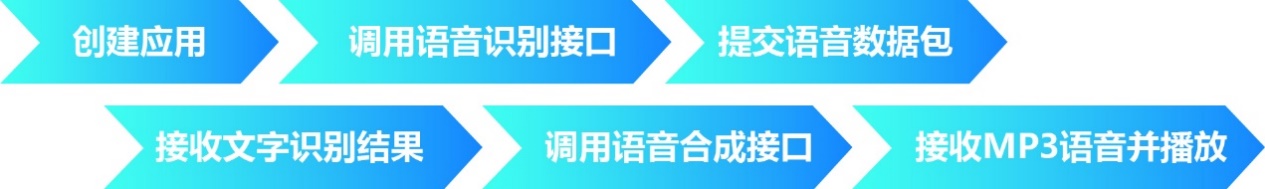
 (3-7)

其中代表输出向量，代表输入向量，代表权重矩阵，代表激活函数，代表偏移向量。

3.6.2 基于百度语音API的语音识别与交互

百度AI云平台适用于语音识别、合成、控制和对话等场景，不同场景有相同的API Key和Secret Key，只要以HTTPS的方式向平台请求服务，就可以得到AI分析结果。百度AI平台调用开放API需要使用OAuth2.0授权，需要在URL里包含AccesssToken参数，此参数可以向授权服务地址发送请求来获取[23]。

为实现本地嵌入式平台与百度云端通信，创建应用并将对应的API Key和Secret Key填写到Python代码中，通过编写程序，调用百度REST API给予的标准HTTP端口，按照JSON数据格式，将采集的语音数据（pcm、wav、amr、m4a等格式，采样率16000、8000，16bit位深，单声道）上传百度AI云平台（语音识别请求地址：<http://vop.baidu.com/server_api>）进行识别，接收文字识别结果，并通过百度AI语音合成接口获得一段MP3进行语音播放，即实现了语音的识别与合成，调用流程如下图3-17所示。



**图3-17 百度API调用流程图**

3.6.3 基于语音模块的语音识别

系统外接LD3320语音模块，通过烧录相应的固件或SDK文件，模块能够实现简单的语音交互，通过将该模块与Arduino的IO口相连接，当司机通过该模块说出相应的语音指令后，模块的某个IO口输出高电平，Arduino通过检测不同IO口的高低电平判断用户的指令，并通过串口将指令信息上传至主机，主机则会进行相应的操作，同时LD3320也配备相应的串口，也可通过串口实现与主机的直接通信。



**图3-18 语音模块实物图**

在使用串口时，其相关配置如下

**表3-3 串口配置**

|  |  |  |  |
| --- | --- | --- | --- |
| 波特率 | 数据位 | 停止位 | 校验位 |
| 9600 | 8 | 1 | None |

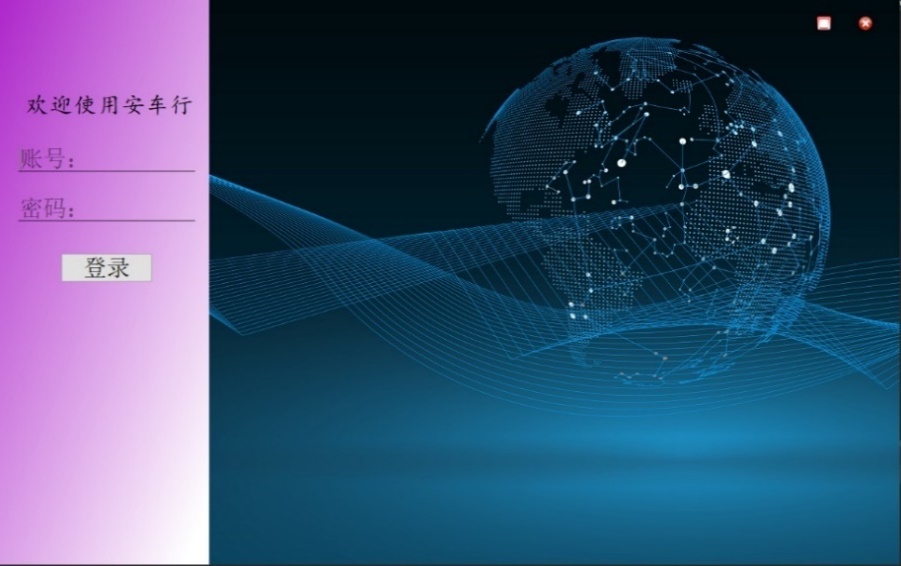
此外LD3320的识别准确率较高，识别距离能够达到5m的距离，并且支持多达上百条语句的的识别，能够很好的应用与车载环境中，UNV-LD3320的产品规格如下表

**表3-4 产品规格说明**

|  |  |  |
| --- | --- | --- |
| 硬  件  介  绍 | 项目 | 说明 |
| 对外接口 | 9个IO口或者（7个IO+一组串口） |
| 工作温度 | -40℃~105℃ |
| 工作电压 | 5V (电流推荐不小于1A) |
| 供电电流 | 不小于150MA |
| 封装尺寸 | 19.5\*22\*2.5mm |
| 喇叭接口 | 板载功放（注意正负极） |
| 咪头接口 | 直接接咪头 |
| 词条 | 唤醒词+150条 |

3.6.4 UI界面

市场上车载触摸屏的交互界面设计的核心是简洁，大方，交互设计的核心是用户，出发点是用户需求，目的是创造和构建用户与产品或服务之间有意义的关系，其本质是研究“人”与“机器”进行信息交流时的各种行为特征、认知以及正确理解、正确使用和便于使用的问题。基于此需求，同时充分显示本系统的各项功能，设计了如下一款交互界面



**图3-19 安车行登录界面**

首先系统配备了登录界面，有效的保障了司机的个人隐私，同时在登录过程中通过摄像头对司机的脸部特征点进行提取，确定相应的疲劳阈值，实现针对不同司机疲劳状态的自适应识别。登录成功后进入到用户的主界面，如下所示:：



**图3-20 安车行主界面**

进入主界面后，三个摄像头都将会自动开启，系统在后台同时对司机的疲劳状态，驾驶行为以及车外路况进行检测，当遇到异常情况时将会进行相应的警告，其中包含弹窗及语音提示，以辅助司机的驾驶，同时在界面上显示当天时间与天气状况。司机还可以通过按下按钮实现各个摄像头图像与检测信息的实时显示。触摸屏界面交互是用户通过触摸控制的方式与屏幕进行互动，是随着互联网＋时代的来临得以广泛使用的交互方式。

3.6.5 音乐播放

系统还具有播放音乐的功能，通过语音或者触摸屏，驾驶员可以实时便捷的播放，暂停，切换音乐，当检测到司机出现愤怒的情绪时，系统也会自动的播放舒缓的音乐安抚驾驶员的情绪。

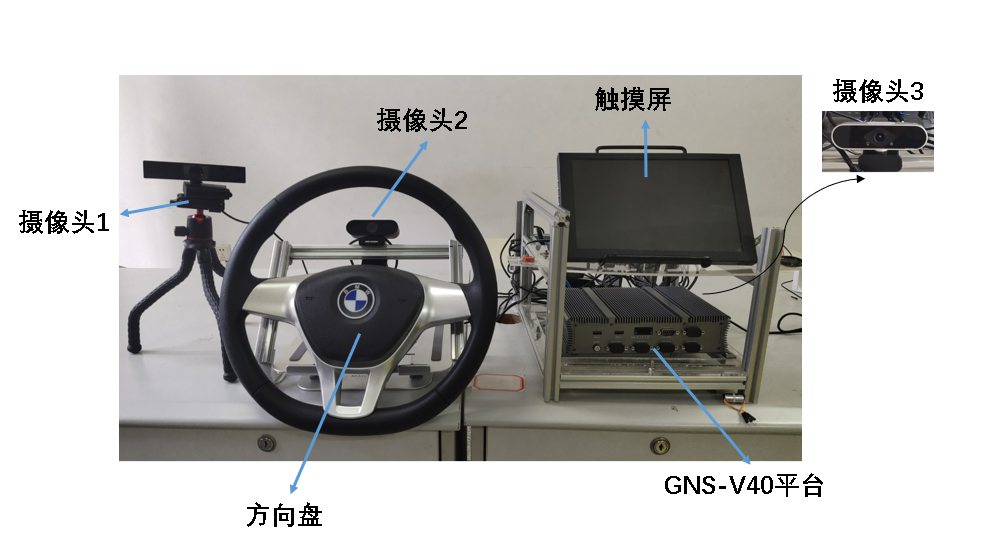
1. 系统测试

4.1 系统测试方案

由于条件有限，本系统测试均在实验室环境中完成。测试过程可分为疲劳测试、危险行为测试、道路安全测试、情感测试和智能交互测试。首先将三目摄像头接入系统平台，进行疲劳和情感检测的测试，分析疲劳检测和情感检测准确率。接着对于危险行为检测进行测试，分析其精度。之后对车外路况检测部分测试，分析其准确度，最后对系统其它功能模块进行测试，分析各模块可行性，测试系统整体性能。

4.2 测试设备

安车行——车载安全驾驶系统（本作品）的测试设备如图4-1所示。



**图4-1 测试设备实物图**

4.3 疲劳检测测试

在实验室中，三位测试者在光线充足条件下依次在距离摄像头约0.2m、0.4m、0.6m处以不同的疲劳状态进行30次测试，随后在光线不足条件下重复一次测试，采用车内第一个摄像头进行视频拍摄记录作为疲劳检测测试数据集。测试统计数据如表4-1所示。

**表4-1 疲劳检测测试成功率数据统计表（CPU）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试对象 | 准确率 | | | | | |
| 光线充足 | | | 光线不足 | | |
| 距离0.2m | 距离0.4m | 距离0.6m | 距离0.2m | 距离0.4m | 距离0.6m |
| 甲 | 90% | 93% | 86% | 87% | 90% | 83% |
| 乙 | 87% | 90% | 89% | 84% | 85% | 81% |
| 丙 | 90% | 95% | 93% | 85% | 90% | 86% |

4.4 情感识别测试

在与疲劳检测相同的实验室环境下，采用车内第一个摄像头进行视频视频拍摄记录作为情感识别测试数据集。测试统计数据如表4-2所示。

**表4-2 情感检测测试成功率数据统计表（CPU）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试对象 | 准确率 | | | | | |
| 光线充足 | | | 光线不足 | | |
| 距离0.2m | 距离0.4m | 距离0.6m | 距离0.2m | 距离0.4m | 距离0.6m |
| 甲 | 90% | 96% | 93% | 80% | 86% | 80% |
| 乙 | 92% | 97% | 94% | 85% | 86% | 89% |
| 丙 | 95% | 95% | 90% | 91% | 93% | 87% |

4.5 危险行为检测测试

在光线充足和光线不足的实验室，三名测试者利用车内第二个摄像头依次对其吸烟、拨打电话的危险行为测试30次，并利用车内酒精传感器对不同浓度的酒精测试30次。测试数据如表4-3所示。

**表4-3 危险行为测试数据统计表（CPU）**

|  |  |  |  |
| --- | --- | --- | --- |
| 测试对象 | 准确率 | | |
| 吸烟 | 拨打电话 | 酒驾 |
| 甲 | 97% | 97% | 95% |
| 乙 | 100% | 98% | 100% |
| 丙 | 98% | 96% | 100% |

4.6路况检测测试

本系统针对车外道路检测，在白天、夜晚上对于普通拥挤的城市道路交通和高速公路对行人、非机动车、机动车分别测试。测试数据如表4-4所示。

**表4-4 路况检测测试成功率数据统计表（CPU）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试对象 | 准确率 | | | |
| 白天 | | 夜晚 | |
| 城市道路 | 高速公路 | 城市道路 | 高速公路 |
| 行人 | 94% | 98% | 90% | 93% |
| 非机动车 | 92% | 无 | 87% | 无 |
| 机动车 | 95% | 98% | 90% | 95% |

4.7智能交互测试

考虑到驾驶员在驾驶环境下可能有较大噪音情况出现，因此系统在安静和嘈杂环境下进行语音交互测试。项目邀请了三位测试者依次测试30次，测试数据如表4-5所示。

**表4-5 智能交互检测测试成功率数据统计表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试对象 | 准确率 | | | |
| 安静 | | 嘈杂 | |
| 语音唤醒 | 语音交互 | 语音唤醒 | 语音交互 |
| 甲 | 98% | 97% | 90% | 95% |
| 乙 | 98% | 100% | 90% | 97% |
| 丙 | 97% | 99% | 87% | 93% |

4.8整体性测试

对本系统的整体功能进行测试，主要是在多线程和单独运行的条件下观察各功能cpu占用率和fps的情况，测试10次，测试数据如表4-6所示。

**表4-6 整体性测试处理速度统计表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试对象 | cpu占有率 | | fps | |
| 单独运行 | 多线程 | 单独运行 | 多线程 |
| 疲劳检测 | 30% | 26% | 64 | 67 |
| 情感识别 | 12% | 10% | 72 | 67 |
| 危险行为 | 13% | 10% | 70 | 72 |
| 路况检测 | 26% | 23% | 70 | 71 |
| 智能交互 | 30% | 25% | - | - |

4.9 结果分析

测试结果表明，系统各模块运行效果良好。对于疲劳检测算法，在硬件平台CPU下准确率约为90%，对于情感识别算法，准确率约为92%，对于危险行为检测，其准确度均在95%以上，对于路况检测，其精度基本在90%以上，且加速后每秒fps均有所提升，展示了硬件平台强大的视觉处理能力。语音交互部分识别精度能够达90%以上。且使用多进程可以同时运行各项功能，提升了cpu的利用率并加快了视频的处理速度。

第五章 总结与展望

5.1 系统特色

（1）多技术融合，稳定高效

系统融入疲劳检测、路况检测、人体姿态估计、语音识别、多媒体、通信等技术，鲁棒性较高，稳定性较好。

（2）关注社会焦点问题

当前交通事故频发，且受社会重点关注，故而设计一款针对车辆安全驾驶的系统非常重要。

（3）硬件简单，成本低廉

系统采用的多为当前较成熟的外设模块，摄像头要求并不是十分严格，配合边缘计算算法，具有较高的性价比。

（4）充分挖掘系统资源提高系统性能

在系统实现时，深度挖掘GNS-V40硬件平台的资源，充分利用AI卡强大的视觉处理加速功能，对程序进行优化，有效降低CPU负载，提高处理速度，提高系统性能。

（5）交叉应用潜力大

本系统主要基于车载摄像头进行设计，旨在车辆安全驾驶领域提供见解，有潜力与其他道路安全产品进行融合，有潜力在道路安全等领域发挥重要作用。

5.2 前景展望

（1）在技术发展趋势上，人工智能技术的发展经历了几十年的起起落落，在一次次高潮与低估的交替中不断前进，近年来以深度学习为代表的AI技术取得了突破性进展，在计算机视觉、自然语言处理、机器人技术方面取得了巨大的进步，“算法+算力+数据”三者互相促进、不断迭代，引发了AI的第三次高潮，深刻地改变了我们的生活，在医疗卫生服务、文教娱乐领域、制造业和服务行业中均得到了广泛应用。

计算机视觉技术是利用摄影机和计算机模拟生物的眼睛对目标进行识别和跟踪，使系统从视频、图片或多维数据中感知并且获取信息，目前人脸识别、指纹识别等在智能安防、身份识别方面取得广泛应用。根据统计，从技术层面看，计算机视觉领域是最受创投圈欢迎的人工智能技术，是各大企业积极布局以及投资者极为看好的方向。

本项目计划融入计算机视觉相关算法与技术，借助Intel平台的高处理能力，基于摄像头等视频采集装置，对驾驶员进行识别与追踪，对一些重要或异常行为进行监测和监督，以此确保驾驶员安全驾驶。随着计算机视觉有关技术的不断发展，相信项目稳定性、可靠性方面均有机会不断提升，预计项目具有较好的技术发展前景。

边缘计算是指在网络边缘进行计算的技术，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。其应用程序在边缘侧发起，产生更快的网络服务响应，满足行业在实时业务、应用智能、安全与隐私保护等方面的基本需求。边缘计算作为继云计算之后新的计算范式，将计算下沉到靠近用户和数据源的网络边缘，提供数据缓存和处理功能，具有低延迟、安全性高、位置感知等特点。

随着人工智能和物联网技术的发展，海量数据需要快速有效的提取和分析，这大大加强了对于边缘计算的需求。未来AI技术、边缘计算和物联网将更加密切进行融合发展，尤其在安防行业视频监控领域的应用。而且随着芯片技术的发展，先进硬件将为人工智能模型的速度提供更高的支持来进行复杂的计算，尤其是前端设备，在安防行业中的应用，需要体积更小、功能更加强大的嵌入式芯片来运行性能更好的算法，用于实时跟踪、面部识别等应用。

目前边缘计算尚处于发展阶段，算法和硬件设施等都在不断进步，在智能驾驶领域具有较好的应用前景。项目融入边缘计算，将为驾驶员提供更加优质安全的服务。

（2）在政策支持上，项目的目标人群为机动车驾驶员，定位为智能化安全驾驶产品。在当前社会背景下，智能驾驶行业具有广阔的市场需求和良好的发展前景，同时国家政策频发，为市场发展提供了有力的催化剂。而项目融入人工智能+边缘计算，在人工智能领域，我国已系统布局，整体部署我国的人工智能发展规划，抢抓人工智能发展的战略机遇，为行业发展带来及时雨，有力推动人工智能产业进步。由此可见，国家政策上为产业发展提供了相应的支持，项目设计内容符合社会发展需要，符合国家战略支持，具有丰富的时代背景和深刻的现实意义。

参考文献

[1]沈霄鹏.共享经济下道德风险及其防控研究[D].杭州: 杭州电子科技大学, 2019.

[2]郝文清.道德风险的防范与化解[J].社会科学家, 2011(5): 98-101.

[3]潘静.“网约车”类共享经济的保险规制路径[J]. 武汉 金融,2017(8):65-69

[4]冯骅,王勇.网约车监管的改革方向: 实施双重监管体系［J］. 企业经济, 2020(2): 139-145.

[5]刘明.网络约租车的侵权责任分担机制[J]. 财经法学, 2016(5):52-57+40.

[6]张新宝.顺风车网络平台的安全保障义务与侵权责任[J]. 法律适用,2018(12):98-104

[7] 樊星,刘占文,林杉等.融合人眼特征与深度学习的疲劳状态检测模型[J/OL].计算机工程：1-17［2021-01-04］.

[8] 徐明.嵌入式驾驶员疲劳检测系统应用软件设计与实现[D].郑州：郑州大学，2014.

[9] 郑伟成,李学伟,刘宏哲等.基于深度学习的疲劳驾驶检测算法[J].计算机工程，2020，46（7）：21-29.

[10] JIANG K，LING F，FENG Z，et al. Why do drivers continue driving while fatigued？ An application of the theory of planned behavior［J］. Transportation Research Part A：Policy and Practice，2017，98：141-149.

[11] https://www.zhihu.com/question/59750782

[12] 闵建亮,蔡铭.基于前额脑电多尺度小波对数能量熵的驾驶疲劳检测分析[J].中国公路学报，2020，33（6）：182-189.

[13] 庄员,戚湧.伪3D卷积神经网络与注意力机制结合的疲劳驾驶检测[J].中国图象图形学报，2021 26 (1);142－153．

[14] https://www.jianshu.com/p/3aa810b35a5d

[15] 李彬，汪诚，吴静,等. 改进YOLOv4算法的航空发动机部件表面缺陷检测[J].激光与光电子学进展, 2021:1-17[2021-04-07]

[16] 李昭慧,张玮良.基于改进YOLOv4算法的疲劳驾驶检测[J].电子测量技术报, 2021.7

[17] https://www.jianshu.com/p/c1c19416c94b

[18] <https://www.csdn.net/tags/MtTakg0sMTAxMDEtYmxvZwO0O0OO0O0O.html>

[19]田广,戚飞虎,朱文佳,毛欣,陈磐君.单目移动拍摄下基于人体部位的行人检测[J].系统仿真学报,2006(10):2906-2910.

[20] 赵宗玉.人工智能技术现状剖析[J].中国安防,2020(03):29-33.

[21] http://www.w3.org/1999/xhtml

[22] <https://baike.so.com/doc/2709039-2860084.html>

[23] 牛可. 智能家居语音识别通用语音AI云平台的设计与实现[D]. 2019.

**附录**

#情绪识别部分

from openvino.inference\_engine import IECore

import numpy as np

import time

import cv2 as cv

emotions = ['neutral', 'happy', 'sad', 'surprise', 'anger']

def face\_emotion\_demo():

ie = IECore()

#加载人脸检测模型

model\_xml = "D:/projects/models/face-detection-0102/FP32/face-detection-0102.xml"

model\_bin = "D:/projects/models/face-detection-0102/FP32/face-detection-0102.bin"

net = ie.read\_network(model=model\_xml, weights=model\_bin)

input\_blob = next(iter(net.input\_info))

out\_blob = next(iter(net.outputs))

#获取输入图像大小

n, c, h, w = net.input\_info[input\_blob].input\_data.shape

#调用摄像头

cap = cv.VideoCapture(0)

exec\_net = ie.load\_network(network=net, device\_name="CPU")

# 加载人脸表情识别模型

em\_xml = "D:/projects/models/emotions-recognition-retail-0003/FP32/emotions-recognition-retail-0003.xml"

em\_bin = "D:/projects/models/emotions-recognition-retail-0003/FP32/emotions-recognition-retail-0003.bin"

#加载神经网络

em\_net = ie.read\_network(model=em\_xml, weights=em\_bin)

#导入输入图片

em\_input\_blob = next(iter(em\_net.input\_info))

em\_out\_blob = next(iter(em\_net.outputs))

en, ec, eh, ew = em\_net.input\_info[em\_input\_blob].input\_data.shape

em\_exec\_net = ie.load\_network(network=em\_net, device\_name="CPU")#CPU加速

while True:

ret, frame = cap.read()

if ret is not True:

break

image = cv.resize(frame, (w, h))

image = image.transpose(2, 0, 1)#图像矩阵变换

inf\_start = time.time()

res = exec\_net.infer(inputs={input\_blob: [image]})#开始模型推理

inf\_end = time.time() - inf\_start#获取推理时间

ih, iw, ic = frame.shape

res = res[out\_blob]

for obj in res[0][0]:

if obj[2] > 0.75:#若置信度大于设定阈值（0.75)

xmin = int(obj[3] \* iw)

ymin = int(obj[4] \* ih)

xmax = int(obj[5] \* iw)

ymax = int(obj[6] \* ih)

if xmin < 0:

xmin = 0

if ymin < 0:

ymin = 0

if xmax >= iw:

xmax = iw - 1

if ymax >= ih:

ymax = ih - 1#限定人脸位置

roi = frame[ymin:ymax, xmin:xmax, :]

roi\_img = cv.resize(roi, (ew, eh))

roi\_img = roi\_img.transpose(2, 0, 1)#图像矩阵逆变换

em\_res = em\_exec\_net.infer(inputs={em\_input\_blob: [roi\_img]})#将获取的人脸图像导入情绪识别模型

prob\_emotion = em\_res[em\_out\_blob].reshape(1, 5)

label\_index = np.argmax(prob\_emotion, 1)#获取情绪识别结果标签

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 255, 255), 2, 8)#框出对应人脸

cv.putText(frame, "infer time(ms): %.3f" % (inf\_end \* 1000), (50, 50), cv.FONT\_HERSHEY\_SIMPLEX, 1.0,

(255, 0, 255),

2, 8)#在图像上显示推理时间

cv.putText(frame, emotions[np.int(label\_index)], (xmin, ymin), cv.FONT\_HERSHEY\_SIMPLEX, 0.55,

(0, 0, 255),

2, 8)#在图像上显示情绪

cv.imshow("Face+emotion Detection", frame)#显示图像

c = cv.waitKey(1)

if c == 27:

break

cv.waitKey(0)

cv.destroyAllWindows()

#登录界面

Import Login as \*

from PyQt5.QtWidgets import QApplication,QMainWindow

from PyQt5 import QtGui, QtCore

class LoginWindow(QMainWindow):#定义登录界面类

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.ui=Login\_MainWindow()

self.ui.setupUi(self)

self.setWindowFlag(QtCore.Qt.FramelessWindowHint)

self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)

self.ui.pushButton.clicked.connect(self.go\_to\_inter)#若用户按下登录键后开始采集

self.show()

def go\_to\_inter(self):#采集用户初始化图像

account=self.ui.lineEdit.text()

password=self.ui.lineEdit\_2.text()

if account=='anchexing' and password=='123456':#若账号密码输入正确

print(main\_camera())#采集用户脸部的疲劳阈值

with open("start.txt", "w") as f:#其它三个进程开启

f.write("1")

InterWindow()#进入安车行主界面

self.close()

else :

pass

#设置UI界面可移动

def mousePressEvent(self, event):

if event.button() == QtCore.Qt.LeftButton and self.isMaximized() == False:

self.m\_flag = True

self.m\_Position = event.globalPos() - self.pos() # 获取鼠标相对窗口的位置

event.accept()

self.setCursor(QtGui.QCursor(QtCore.Qt.OpenHandCursor)) # 更改鼠标图标

def mouseMoveEvent(self, mouse\_event):

if QtCore.Qt.LeftButton and self.m\_flag:

self.move(mouse\_event.globalPos() - self.m\_Position) # 更改窗口位置

mouse\_event.accept()

def mouseReleaseEvent(self, mouse\_event):

self.m\_flag = False

self.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))

#主界面

class InterWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.r = requests.get('http://www.weather.com.cn/data/sk/101110101.html')#获取当地天气信息

self.r.encoding = 'utf-8'

self.ser = serial.Serial(port="COM9", baudrate=9600,stopbits=1,bytesize=8,parity='N',timeout=0.5)#定义串口通信相关参数

while (self.ser.isOpen() == 0):

print('打开串口失败')

self.ser.open()

self.alcohol=False

self.count=0

self.ui=Ui\_MainWindow()

self.ui.setupUi(self)

self.setWindowFlag(QtCore.Qt.FramelessWindowHint)

self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)

#定义各种按钮跳转的函数

self.ui.pushButton\_tired.clicked.connect(change\_tired)

self.ui.pushButton\_road.clicked.connect(change\_outside)

self.ui.pushButton\_action.clicked.connect(actdetection)

self.ui.pushButton\_exit.clicked.connect(exitanger)

self.ui.pushButton\_music.clicked.connect(playmusic)

self.ui.pushButton\_chick.clicked.connect(changemusic)

self.ui.pushButton\_emotion.clicked.connect(emotionstart)

self.i = datetime.datetime.now()#获取当前的时间信息

self.ui.pushButton\_weather.setText('西安\n'+str(self.i.month)+'月'+str(self.i.day)+'日\n'+self.r.json()['weatherinfo']['temp']+'°C '+self.r.json()['weatherinfo']['WD'])#显示时间与天气信息

timer=QtCore.QTimer()#设置中断定时器，每隔300ms进入一次中断

self.timer=timer

self.timer.start(300)

self.timer.timeout.connect(self.clicked\_button)

self.show()

def clicked\_button(self):#中断服务函数

self.i=datetime.datetime.now()

date=str(self.i.hour)

self.ui.label.setText(str(self.i.hour)+':'+str(self.i.minute))#实时更新当前时间

self.music()#语音模块识别

with open("people\_alarm.txt","r") as f:#若检测到前方行人过近

if f.readline()=='1':

PeopleWindow()

with open("car\_alarm.txt","r") as f:#若检测到前方车辆过近

if f.readline()=='1':

CarWindow()

with open("tired\_alarm.txt","r") as f:#若检测到疲劳驾驶

if f.readline()=='1':

TiredWindow()

with open("action\_alarm.txt","r") as f: #若检测危险行为

if f.readline()=='1':

WarnWindow()

def music(self):

self.ser.flushInput()#清空之前串口缓存区的数据

self.num = self.ser.readline()#读取当前获取的最新串口信息

self.count+=1

print(self.count)

if self.count>=2 :#除去第一个标识符

if self.num.decode('utf-8') == '9\r\n':

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\1.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

elif self.num.decode('utf-8') == '8\r\n':

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\2.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

elif self.num.decode('utf-8') == '7\r\n':

mixer.music.stop() # 结束音乐播放

elif self.num.decode('utf-8') == 'n\r\n':

self.ui.pushButton\_alcohol.setText('酒精检测\n正常')

elif self.num.decode('utf-8') == 'w\r\n':

self.ui.pushButton\_alcohol.setText('酒精检测\n浓度过高')

#按下疲劳检测按键后跳转的函数 ，其它按钮与此类似

def change\_tired():

global tired\_click

tired\_click=1-tired\_click

if tired\_click==1:

with open("tired\_change.txt", "w") as f:

f.write("1")

else :

with open("tired\_change.txt", "w") as f:

f.write("0")

#按下播放音乐按钮

def playmusic():

global playstate

if playstate==0:

print("play!")

playstate=1

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\1.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

elif playstate==1 or playstate==2:

playstate=0

mixer.music.stop() # 结束音乐播放

#切换音乐

def changemusic():

global playstate

if playstate==2:

playstate=1

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\1.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

elif playstate==1:

playstate=2

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\2.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

#危险行为弹窗 其它弹窗与此类似

class WarnWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.ui = Warn\_MainWindow()#加载危险驾驶的UI界面

self.ui.setupUi(self)

self.setWindowFlag(QtCore.Qt.FramelessWindowHint)

self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)

self.ui.label.setText('请勿危险驾驶')

self.ui.pushButton.clicked.connect(lambda: self.ui.label.setText('请勿危险驾驶'))

timer = QtCore.QTimer()

self.timer=timer

self.timer.start(500)

self.timer.timeout.connect(self.warn\_time)#每隔500ms检测一次

self.show()

def warn\_time(self):

with open("warn\_alarm.txt","r") as f:

if f.readline()=='0':#若不存在危险行为则停止弹窗报警

self.close()

#定义UI界面线程

def GUI(x):

app = QApplication(sys.argv)

win=LoginWindow()

app.exec\_()

#初始化与主函数

if \_\_name\_\_=='\_\_main\_\_':

#初始化多进程所共享的内存文件

with open("start.txt", "w") as f:

f.write("0")

with open("people\_alarm.txt", "w") as f:

f.write("0")

with open("car\_alarm.txt", "w") as f:

f.write("0")

with open("action\_alarm.txt", "w") as f:

f.write("0")

with open("outside\_change.txt", "w") as f:

f.write("0")

with open("tired\_change.txt.txt", "w") as f:

f.write("0")

with open("emotion.txt", "w") as f:

f.write("0")

with open("ready.txt","w") as f:

f.write("0")

#开启进程池，打开四路进程

pool = multiprocessing.Pool(processes=4)

#分配进程

pool.apply\_async(outsidetection, (1,))

pool.apply\_async(GUI, (2,))

pool.apply\_async(tiredetection, (3,))

pool.apply\_async(actiondetection,(4,))

#等待进程结束或关闭

pool.close()

pool.join()

#登录界面的python源码

from PyQt5 import QtCore, QtGui, QtWidgets

import res\_rc

class Login\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(1277, 939)

font = QtGui.QFont()

font.setPointSize(25)

MainWindow.setFont(font)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(0, 0, 300, 800))

self.label.setStyleSheet("background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:0.898, y2:0.977273, stop:0 rgba(173, 37, 202, 255), stop:1 rgba(255, 255, 255, 255));\n"

"")

self.label.setText("")

self.label.setObjectName("label")

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

self.label\_2.setGeometry(QtCore.QRect(300, 0, 980, 800))

self.label\_2.setStyleSheet("background-image: url(:/image/image/ground5.png);")

self.label\_2.setText("")

self.label\_2.setObjectName("label\_2")

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setGeometry(QtCore.QRect(40, 120, 241, 61))

font = QtGui.QFont()

font.setFamily("楷体")

font.setPointSize(25)

self.label\_3.setFont(font)

self.label\_3.setObjectName("label\_3")

self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit.setGeometry(QtCore.QRect(30, 210, 250, 35))

font = QtGui.QFont()

font.setPointSize(25)

self.lineEdit.setFont(font)

self.lineEdit.setStyleSheet("border:none;\n"

"border-bottom:2px solid rgba(0,0,0,100);\n"

"background-color: rgb(0, 0, 0 ,0 );\n"

"")

self.lineEdit.setObjectName("lineEdit")

self.lineEdit\_2 = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit\_2.setGeometry(QtCore.QRect(30, 280, 250, 35))

font = QtGui.QFont()

font.setPointSize(25)

self.lineEdit\_2.setFont(font)

self.lineEdit\_2.setStyleSheet("border:none;\n"

"border-bottom:2px solid rgba(0,0,0,100);\n"

"background-color: rgb(0,0,0,0);\n"

"")

self.lineEdit\_2.setEchoMode(QtWidgets.QLineEdit.Password)

self.lineEdit\_2.setObjectName("lineEdit\_2")

self.frame = QtWidgets.QFrame(self.centralwidget)

self.frame.setGeometry(QtCore.QRect(1120, 0, 161, 71))

self.frame.setStyleSheet("background-color: rgb(255, 255, 255,0);\n"

"")

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.frame)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setSpacing(0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.pushButton\_2 = QtWidgets.QPushButton(self.frame)

self.pushButton\_2.setMaximumSize(QtCore.QSize(20, 20))

self.pushButton\_2.setStyleSheet("border-image: url(:/icon/icon/19.gif);")

self.pushButton\_2.setText("")

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap(":/icon/19.gif"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_2.setIcon(icon)

self.pushButton\_2.setObjectName("pushButton\_2")

self.horizontalLayout.addWidget(self.pushButton\_2)

self.pushButton\_3 = QtWidgets.QPushButton(self.frame)

self.pushButton\_3.setMaximumSize(QtCore.QSize(20, 20))

self.pushButton\_3.setStyleSheet("border-image: url(:/icon/icon/Cancel.ico);\n"

"")

self.pushButton\_3.setText("")

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap(":/icon/Cancel.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_3.setIcon(icon1)

self.pushButton\_3.setObjectName("pushButton\_3")

self.horizontalLayout.addWidget(self.pushButton\_3)

self.pushButton = QtWidgets.QPushButton(self.centralwidget)

self.pushButton.setGeometry(QtCore.QRect(90, 360, 130, 41))

self.pushButton.setObjectName("pushButton")

MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

self.pushButton\_3.clicked.connect(MainWindow.close)

self.pushButton\_2.clicked.connect(MainWindow.showMaximized)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.label\_3.setText(\_translate("MainWindow", "欢迎使用安车行"))

self.lineEdit.setPlaceholderText(\_translate("MainWindow", "账号："))

self.lineEdit\_2.setPlaceholderText(\_translate("MainWindow", "密码："))

self.pushButton.setText(\_translate("MainWindow", "登录"))

#主界面的python源码

from PyQt5 import QtCore, QtGui, QtWidgets

import uircc1\_rc

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(1298, 840)

MainWindow.setIconSize(QtCore.QSize(32, 32))

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)

self.horizontalLayout.setObjectName("horizontalLayout")

self.frame = QtWidgets.QFrame(self.centralwidget)

self.frame.setMinimumSize(QtCore.QSize(1280, 800))

self.frame.setMaximumSize(QtCore.QSize(1280, 800))

self.frame.setStyleSheet("#frame{\n"

" \n"

" background-image: url(:/image/R-C (1).jpg);\n"

" border-radius:25px;\n"

" \n"

"}")

self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame.setObjectName("frame")

self.verticalLayout = QtWidgets.QVBoxLayout(self.frame)

self.verticalLayout.setContentsMargins(0, 0, 0, 0)

self.verticalLayout.setSpacing(0)

self.verticalLayout.setObjectName("verticalLayout")

self.frame\_2 = QtWidgets.QFrame(self.frame)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(1)

sizePolicy.setHeightForWidth(self.frame\_2.sizePolicy().hasHeightForWidth())

self.frame\_2.setSizePolicy(sizePolicy)

self.frame\_2.setStyleSheet("\n"

"#frame\_2{\n"

" border-top-left-radius:25px;\n"

" border-top-right-radius:25px;\n"

" background-color: rgb(255, 255, 255,75);\n"

"}")

self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_2.setLineWidth(0)

self.frame\_2.setObjectName("frame\_2")

self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.frame\_2)

self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_2.setSpacing(0)

self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")

self.frame\_7 = QtWidgets.QFrame(self.frame\_2)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(3)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_7.sizePolicy().hasHeightForWidth())

self.frame\_7.setSizePolicy(sizePolicy)

self.frame\_7.setStyleSheet("background-color: rgb(255, 255, 255,0);")

self.frame\_7.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_7.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_7.setObjectName("frame\_7")

self.horizontalLayout\_8 = QtWidgets.QHBoxLayout(self.frame\_7)

self.horizontalLayout\_8.setObjectName("horizontalLayout\_8")

self.frame\_6 = QtWidgets.QFrame(self.frame\_7)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_6.sizePolicy().hasHeightForWidth())

self.frame\_6.setSizePolicy(sizePolicy)

self.frame\_6.setStyleSheet("QPushButton{\n"

" border:none;\n"

"}\n"

"\n"

"QPushButton:hover{\n"

" padding-bottom:5px;\n"

"}")

self.frame\_6.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_6.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_6.setObjectName("frame\_6")

self.horizontalLayout\_3 = QtWidgets.QHBoxLayout(self.frame\_6)

self.horizontalLayout\_3.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_3.setSpacing(0)

self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")

self.pushButton\_back = QtWidgets.QPushButton(self.frame\_6)

font = QtGui.QFont()

font.setPointSize(9)

self.pushButton\_back.setFont(font)

self.pushButton\_back.setStyleSheet("border:none;")

self.pushButton\_back.setText("")

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap(":/icon/left.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_back.setIcon(icon)

self.pushButton\_back.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_back.setObjectName("pushButton\_back")

self.horizontalLayout\_3.addWidget(self.pushButton\_back)

self.pushButton\_home = QtWidgets.QPushButton(self.frame\_6)

font = QtGui.QFont()

font.setPointSize(9)

self.pushButton\_home.setFont(font)

self.pushButton\_home.setStyleSheet("border:none;")

self.pushButton\_home.setText("")

icon1 = QtGui.QIcon()

icon1.addPixmap(QtGui.QPixmap(":/icon/home.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_home.setIcon(icon1)

self.pushButton\_home.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_home.setObjectName("pushButton\_home")

self.horizontalLayout\_3.addWidget(self.pushButton\_home)

self.horizontalLayout\_8.addWidget(self.frame\_6)

self.label = QtWidgets.QLabel(self.frame\_7)

self.label.setMinimumSize(QtCore.QSize(800, 20))

self.label.setMaximumSize(QtCore.QSize(800, 16777215))

font = QtGui.QFont()

font.setFamily("Yu Gothic")

font.setPointSize(30)

self.label.setFont(font)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.horizontalLayout\_8.addWidget(self.label)

self.frame\_8 = QtWidgets.QFrame(self.frame\_7)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_8.sizePolicy().hasHeightForWidth())

self.frame\_8.setSizePolicy(sizePolicy)

self.frame\_8.setStyleSheet("QPushButton{\n"

" border:none;\n"

"}\n"

"\n"

"QPushButton:hover{\n"

" padding-bottom:5px;\n"

"}\n"

"")

self.frame\_8.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_8.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_8.setObjectName("frame\_8")

self.horizontalLayout\_4 = QtWidgets.QHBoxLayout(self.frame\_8)

self.horizontalLayout\_4.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_4.setSpacing(0)

self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")

self.pushButton\_data = QtWidgets.QPushButton(self.frame\_8)

self.pushButton\_data.setStyleSheet("border:none;")

self.pushButton\_data.setText("")

icon2 = QtGui.QIcon()

icon2.addPixmap(QtGui.QPixmap(":/icon/statistics.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_data.setIcon(icon2)

self.pushButton\_data.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_data.setObjectName("pushButton\_data")

self.horizontalLayout\_4.addWidget(self.pushButton\_data)

self.pushButton\_setting = QtWidgets.QPushButton(self.frame\_8)

self.pushButton\_setting.setStyleSheet("border:none;")

self.pushButton\_setting.setText("")

icon3 = QtGui.QIcon()

icon3.addPixmap(QtGui.QPixmap(":/icon/gear.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_setting.setIcon(icon3)

self.pushButton\_setting.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_setting.setObjectName("pushButton\_setting")

self.horizontalLayout\_4.addWidget(self.pushButton\_setting)

self.pushButton\_close = QtWidgets.QPushButton(self.frame\_8)

self.pushButton\_close.setStyleSheet("border:none;")

self.pushButton\_close.setText("")

icon4 = QtGui.QIcon()

icon4.addPixmap(QtGui.QPixmap(":/icon/delete.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_close.setIcon(icon4)

self.pushButton\_close.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_close.setObjectName("pushButton\_close")

self.horizontalLayout\_4.addWidget(self.pushButton\_close)

self.horizontalLayout\_8.addWidget(self.frame\_8)

self.horizontalLayout\_2.addWidget(self.frame\_7)

self.verticalLayout.addWidget(self.frame\_2)

self.frame\_3 = QtWidgets.QFrame(self.frame)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(10)

sizePolicy.setHeightForWidth(self.frame\_3.sizePolicy().hasHeightForWidth())

self.frame\_3.setSizePolicy(sizePolicy)

self.frame\_3.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_3.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_3.setLineWidth(0)

self.frame\_3.setObjectName("frame\_3")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.frame\_3)

self.verticalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.verticalLayout\_2.setSpacing(0)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.frame\_4 = QtWidgets.QFrame(self.frame\_3)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(1)

sizePolicy.setHeightForWidth(self.frame\_4.sizePolicy().hasHeightForWidth())

self.frame\_4.setSizePolicy(sizePolicy)

self.frame\_4.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_4.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_4.setObjectName("frame\_4")

self.horizontalLayout\_10 = QtWidgets.QHBoxLayout(self.frame\_4)

self.horizontalLayout\_10.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_10.setSpacing(0)

self.horizontalLayout\_10.setObjectName("horizontalLayout\_10")

self.frame\_14 = QtWidgets.QFrame(self.frame\_4)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(4)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_14.sizePolicy().hasHeightForWidth())

self.frame\_14.setSizePolicy(sizePolicy)

self.frame\_14.setStyleSheet("")

self.frame\_14.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_14.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_14.setObjectName("frame\_14")

self.horizontalLayout\_11 = QtWidgets.QHBoxLayout(self.frame\_14)

self.horizontalLayout\_11.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout\_11.setSpacing(0)

self.horizontalLayout\_11.setObjectName("horizontalLayout\_11")

self.frame\_15 = QtWidgets.QFrame(self.frame\_14)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_15.sizePolicy().hasHeightForWidth())

self.frame\_15.setSizePolicy(sizePolicy)

self.frame\_15.setStyleSheet("image: url(:/image/校标1.png);")

self.frame\_15.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_15.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_15.setObjectName("frame\_15")

self.horizontalLayout\_11.addWidget(self.frame\_15)

self.frame\_16 = QtWidgets.QFrame(self.frame\_14)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(2)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_16.sizePolicy().hasHeightForWidth())

self.frame\_16.setSizePolicy(sizePolicy)

self.frame\_16.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_16.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_16.setObjectName("frame\_16")

self.horizontalLayout\_11.addWidget(self.frame\_16)

self.horizontalLayout\_10.addWidget(self.frame\_14)

self.frame\_9 = QtWidgets.QFrame(self.frame\_4)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_9.sizePolicy().hasHeightForWidth())

self.frame\_9.setSizePolicy(sizePolicy)

self.frame\_9.setMinimumSize(QtCore.QSize(80, 0))

self.frame\_9.setStyleSheet("QPushButton{\n"

" border:none;\n"

"}\n"

"QPushButton:hover{\n"

" padding-bottom:5px;\n"

"}")

self.frame\_9.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_9.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_9.setObjectName("frame\_9")

self.horizontalLayout\_5 = QtWidgets.QHBoxLayout(self.frame\_9)

self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")

self.pushButton\_message = QtWidgets.QPushButton(self.frame\_9)

self.pushButton\_message.setText("")

icon5 = QtGui.QIcon()

icon5.addPixmap(QtGui.QPixmap(":/icon/bubble.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_message.setIcon(icon5)

self.pushButton\_message.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_message.setObjectName("pushButton\_message")

self.horizontalLayout\_5.addWidget(self.pushButton\_message)

self.pushButton\_file = QtWidgets.QPushButton(self.frame\_9)

self.pushButton\_file.setText("")

icon6 = QtGui.QIcon()

icon6.addPixmap(QtGui.QPixmap(":/icon/folder.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton\_file.setIcon(icon6)

self.pushButton\_file.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_file.setObjectName("pushButton\_file")

self.horizontalLayout\_5.addWidget(self.pushButton\_file)

self.horizontalLayout\_10.addWidget(self.frame\_9)

self.verticalLayout\_2.addWidget(self.frame\_4)

self.frame\_5 = QtWidgets.QFrame(self.frame\_3)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(6)

sizePolicy.setHeightForWidth(self.frame\_5.sizePolicy().hasHeightForWidth())

self.frame\_5.setSizePolicy(sizePolicy)

self.frame\_5.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_5.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_5.setObjectName("frame\_5")

self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.frame\_5)

self.verticalLayout\_3.setContentsMargins(5, 5, 5, 5)

self.verticalLayout\_3.setSpacing(5)

self.verticalLayout\_3.setObjectName("verticalLayout\_3")

self.frame\_10 = QtWidgets.QFrame(self.frame\_5)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(4)

sizePolicy.setHeightForWidth(self.frame\_10.sizePolicy().hasHeightForWidth())

self.frame\_10.setSizePolicy(sizePolicy)

self.frame\_10.setStyleSheet("QPushButton{\n"

" border-radius:15px;\n"

"}\n"

"QPushButton:hover{\n"

" padding-bottom:14px;\n"

"}")

self.frame\_10.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_10.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_10.setObjectName("frame\_10")

self.horizontalLayout\_6 = QtWidgets.QHBoxLayout(self.frame\_10)

self.horizontalLayout\_6.setContentsMargins(15, 15, 15, 15)

self.horizontalLayout\_6.setSpacing(25)

self.horizontalLayout\_6.setObjectName("horizontalLayout\_6")

self.pushButton\_weather = QtWidgets.QPushButton(self.frame\_10)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(5)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_weather.sizePolicy().hasHeightForWidth())

self.pushButton\_weather.setSizePolicy(sizePolicy)

self.pushButton\_weather.setMinimumSize(QtCore.QSize(298, 120))

self.pushButton\_weather.setMaximumSize(QtCore.QSize(16777215, 1000000))

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

font.setBold(True)

font.setItalic(False)

font.setWeight(75)

font.setStyleStrategy(QtGui.QFont.PreferDefault)

self.pushButton\_weather.setFont(font)

self.pushButton\_weather.setLayoutDirection(QtCore.Qt.LeftToRight)

self.pushButton\_weather.setStyleSheet("background-color: rgb(163, 147, 255);\n"

"color: rgb(255, 255, 255);\n"

"text-alignment:left;")

self.pushButton\_weather.setIconSize(QtCore.QSize(5, 5))

self.pushButton\_weather.setObjectName("pushButton\_weather")

self.horizontalLayout\_6.addWidget(self.pushButton\_weather)

self.pushButton\_audio = QtWidgets.QPushButton(self.frame\_10)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(3)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_audio.sizePolicy().hasHeightForWidth())

self.pushButton\_audio.setSizePolicy(sizePolicy)

self.pushButton\_audio.setMinimumSize(QtCore.QSize(0, 120))

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

font.setBold(False)

font.setWeight(50)

self.pushButton\_audio.setFont(font)

self.pushButton\_audio.setStyleSheet("background-color: rgb(255, 38, 78);\n"

"color: rgb(85, 255, 255);\n"

"image: url(:/icon/bubble.ico);\n"

"")

self.pushButton\_audio.setObjectName("pushButton\_audio")

self.horizontalLayout\_6.addWidget(self.pushButton\_audio)

self.pushButton\_tired = QtWidgets.QPushButton(self.frame\_10)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(5)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_tired.sizePolicy().hasHeightForWidth())

self.pushButton\_tired.setSizePolicy(sizePolicy)

self.pushButton\_tired.setMinimumSize(QtCore.QSize(0, 120))

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_tired.setFont(font)

self.pushButton\_tired.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))

self.pushButton\_tired.setStyleSheet("background-color: rgb(255, 195, 111);\n"

"color: rgb(255, 255, 255);\n"

"image: url(:/icon/block.ico);")

self.pushButton\_tired.setIconSize(QtCore.QSize(32, 32))

self.pushButton\_tired.setObjectName("pushButton\_tired")

self.horizontalLayout\_6.addWidget(self.pushButton\_tired)

self.frame\_12 = QtWidgets.QFrame(self.frame\_10)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(3)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.frame\_12.sizePolicy().hasHeightForWidth())

self.frame\_12.setSizePolicy(sizePolicy)

self.frame\_12.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_12.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_12.setObjectName("frame\_12")

self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.frame\_12)

self.verticalLayout\_4.setObjectName("verticalLayout\_4")

self.pushButton\_emotion = QtWidgets.QPushButton(self.frame\_12)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(1)

sizePolicy.setHeightForWidth(self.pushButton\_emotion.sizePolicy().hasHeightForWidth())

self.pushButton\_emotion.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_emotion.setFont(font)

self.pushButton\_emotion.setStyleSheet("background-color: rgb(0, 255, 127);")

self.pushButton\_emotion.setObjectName("pushButton\_emotion")

self.verticalLayout\_4.addWidget(self.pushButton\_emotion)

self.frame\_13 = QtWidgets.QFrame(self.frame\_12)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(1)

sizePolicy.setHeightForWidth(self.frame\_13.sizePolicy().hasHeightForWidth())

self.frame\_13.setSizePolicy(sizePolicy)

self.frame\_13.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_13.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_13.setObjectName("frame\_13")

self.horizontalLayout\_9 = QtWidgets.QHBoxLayout(self.frame\_13)

self.horizontalLayout\_9.setObjectName("horizontalLayout\_9")

self.pushButton\_music = QtWidgets.QPushButton(self.frame\_13)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_music.sizePolicy().hasHeightForWidth())

self.pushButton\_music.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(20)

self.pushButton\_music.setFont(font)

self.pushButton\_music.setStyleSheet("background-color: rgb(0, 255, 255);")

self.pushButton\_music.setObjectName("pushButton\_music")

self.horizontalLayout\_9.addWidget(self.pushButton\_music)

self.pushButton\_chick = QtWidgets.QPushButton(self.frame\_13)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_chick.sizePolicy().hasHeightForWidth())

self.pushButton\_chick.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(20)

self.pushButton\_chick.setFont(font)

self.pushButton\_chick.setStyleSheet("background-color: rgb(0, 255, 255);")

self.pushButton\_chick.setObjectName("pushButton\_chick")

self.horizontalLayout\_9.addWidget(self.pushButton\_chick)

self.verticalLayout\_4.addWidget(self.frame\_13)

self.horizontalLayout\_6.addWidget(self.frame\_12)

self.pushButton\_audio.raise\_()

self.pushButton\_tired.raise\_()

self.pushButton\_weather.raise\_()

self.frame\_12.raise\_()

self.verticalLayout\_3.addWidget(self.frame\_10)

self.frame\_11 = QtWidgets.QFrame(self.frame\_5)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Expanding)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(3)

sizePolicy.setHeightForWidth(self.frame\_11.sizePolicy().hasHeightForWidth())

self.frame\_11.setSizePolicy(sizePolicy)

self.frame\_11.setStyleSheet("QPushButton{\n"

" border-radius:15px;\n"

"}\n"

"QPushButton:hover{\n"

" padding-bottom:15px;\n"

"}")

self.frame\_11.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.frame\_11.setFrameShadow(QtWidgets.QFrame.Raised)

self.frame\_11.setObjectName("frame\_11")

self.horizontalLayout\_7 = QtWidgets.QHBoxLayout(self.frame\_11)

self.horizontalLayout\_7.setContentsMargins(15, 15, 15, 15)

self.horizontalLayout\_7.setSpacing(25)

self.horizontalLayout\_7.setObjectName("horizontalLayout\_7")

self.pushButton\_alcohol = QtWidgets.QPushButton(self.frame\_11)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_alcohol.sizePolicy().hasHeightForWidth())

self.pushButton\_alcohol.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_alcohol.setFont(font)

self.pushButton\_alcohol.setStyleSheet("background-color: rgb(255, 181, 78);\n"

"color: rgb(255, 255, 255);")

self.pushButton\_alcohol.setObjectName("pushButton\_alcohol")

self.horizontalLayout\_7.addWidget(self.pushButton\_alcohol)

self.pushButton\_action = QtWidgets.QPushButton(self.frame\_11)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(2)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_action.sizePolicy().hasHeightForWidth())

self.pushButton\_action.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_action.setFont(font)

self.pushButton\_action.setStyleSheet("background-color: rgb(170, 170, 127);\n"

"image: url(:/icon/Danger.ico);\n"

"color: rgb(255, 255, 255);")

self.pushButton\_action.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_action.setObjectName("pushButton\_action")

self.horizontalLayout\_7.addWidget(self.pushButton\_action)

self.pushButton\_road = QtWidgets.QPushButton(self.frame\_11)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(2)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_road.sizePolicy().hasHeightForWidth())

self.pushButton\_road.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_road.setFont(font)

self.pushButton\_road.setStyleSheet("background-color: rgb(127, 62, 63);\n"

"image: url(:/icon/Target1.ico);\n"

"color: rgb(255, 255, 255);")

self.pushButton\_road.setIconSize(QtCore.QSize(30, 30))

self.pushButton\_road.setObjectName("pushButton\_road")

self.horizontalLayout\_7.addWidget(self.pushButton\_road)

self.pushButton\_exit = QtWidgets.QPushButton(self.frame\_11)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(1)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton\_exit.sizePolicy().hasHeightForWidth())

self.pushButton\_exit.setSizePolicy(sizePolicy)

font = QtGui.QFont()

font.setFamily("黑体")

font.setPointSize(25)

self.pushButton\_exit.setFont(font)

self.pushButton\_exit.setStyleSheet("background-color: rgb(70, 73, 255);\n"

"color: rgb(255, 255, 255);")

self.pushButton\_exit.setObjectName("pushButton\_exit")

self.horizontalLayout\_7.addWidget(self.pushButton\_exit)

self.verticalLayout\_3.addWidget(self.frame\_11)

self.verticalLayout\_2.addWidget(self.frame\_5)

self.verticalLayout.addWidget(self.frame\_3)

self.horizontalLayout.addWidget(self.frame)

MainWindow.setCentralWidget(self.centralwidget)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

self.pushButton\_close.clicked.connect(MainWindow.close)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

self.label.setText(\_translate("MainWindow", "13:41"))

self.pushButton\_weather.setText(\_translate("MainWindow", "西安\n"

"晴\n"

"7月20日 星期二"))

self.pushButton\_audio.setText(\_translate("MainWindow", "\n"

"\n"

"\n"

"\n"

"语音助手"))

self.pushButton\_tired.setToolTip(\_translate("MainWindow", "<html><head/><body><p><img src=\":/icon/Coffe.ico\"/> 疲劳驾驶</p></body></html>"))

self.pushButton\_tired.setText(\_translate("MainWindow", "\n"

"\n"

"\n"

"\n"

"疲劳检测"))

self.pushButton\_emotion.setText(\_translate("MainWindow", "路怒检测"))

self.pushButton\_music.setText(\_translate("MainWindow", "播放\n"

"/\n"

"暂停"))

self.pushButton\_chick.setText(\_translate("MainWindow", "下一首"))

self.pushButton\_alcohol.setText(\_translate("MainWindow", "酒精检测"))

self.pushButton\_action.setText(\_translate("MainWindow", "\n"

"\n"

"\n"

"危险行为"))

self.pushButton\_road.setText(\_translate("MainWindow", "\n"

"\n"

"\n"

"路况检测"))

self.pushButton\_exit.setText(\_translate("MainWindow", "退出"))

#路况检测部分

from openvino.inference\_engine import IECore

import time

import cv2 as cv

import wave

import pyaudio

import win32com.client

import pygame

import time

import numpy as np

import os

import serial #导入串口通信库

from playsound import playsound

from pygame import mixer

# 语音合成输出

def speak(s):

print("-->" + s)

win32com.client.Dispatch("SAPI.SpVoice").Speak(s)

def ssd\_video\_demo():

ie = IECore()

for device in ie.available\_devices:

print(device)

ser = serial.Serial(port="COM9", baudrate=9600)

#开始串口通信

ser.port = 'COM9'

ser.baudrate = 9600

ser.stopbits = 1

ser.bytesize = 8

ser.parity = 'N'

while (ser.isOpen() == 0):

print('打开串口失败')

ser.open()

#使用模型

model\_xml = "C:/Program Files (x86)/Intel/openvino\_2021.4.752/deployment\_tools/open\_model\_zoo/tools/downloader/intel/person-vehicle-bike-detection-2001/FP32/person-vehicle-bike-detection-2001.xml"

model\_bin = "C:/Program Files (x86)/Intel/openvino\_2021.4.752/deployment\_tools/open\_model\_zoo/tools/downloader/intel/person-vehicle-bike-detection-2001/FP32/person-vehicle-bike-detection-2001.bin"

net = ie.read\_network(model=model\_xml, weights=model\_bin)

input\_blob = next(iter(net.input\_info))

out\_blob = next(iter(net.outputs))

print(out\_blob)

n, c, h, w = net.input\_info[input\_blob].input\_data.shape

print(n, c, h, w)

#处理视频

cap = cv.VideoCapture(0, cv.CAP\_DSHOW)

exec\_net = ie.load\_network(network=net, device\_name="CPU")

yanshi=0

while True:

ret, frame = cap.read()#每帧

# frame = cv.GaussianBlur(frame, (3, 3), 0, 0)

if ret is not True:

break

image = cv.resize(frame, (w, h))

image = image.transpose(2, 0, 1)

inf\_start = time.time()

res = exec\_net.infer(inputs={input\_blob: [image]})

inf\_end = time.time() - inf\_start

ih, iw, ic = frame.shape

res = res[out\_blob]

s = 0

i=0

j=0

m=0

yanshi=yanshi+1

for obj in res[0][0]:#每个框

if obj[1]==0:#vehicle

if obj[2] > 0.5:

xmin = int(obj[3] \* iw)

ymin = int(obj[4] \* ih)

xmax = int(obj[5] \* iw)

ymax = int(obj[6] \* ih)

s = ymax - ymin

n = 0.5 \* ih/ s

if n < 1:#标记车辆距离小于一米

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 2, 8)#红色

i=1

if 1 < n :

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 0, 0), 2, 8)#蓝色

cv.putText(frame, str(obj[2]), (xmin, ymin), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1, 8)

if obj[1] == 1: # 人

if obj[2] > 0.5:

xmin = int(obj[3] \* iw)

ymin = int(obj[4] \* ih)

xmax = int(obj[5] \* iw)

ymax = int(obj[6] \* ih)

s = ymax - ymin

n = 1.0 \* ih/ s

if n < 5:#标记行人距离小于五米

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 2, 8)

j=1

if 5 < n :

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 0, 0), 2, 8)

cv.putText(frame, str(obj[2]), (xmin, ymin), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1, 8)

if obj[1] == 2: # 自行车

if obj[2] > 0.5:

xmin = int(obj[3] \* iw)

ymin = int(obj[4] \* ih)

xmax = int(obj[5] \* iw)

ymax = int(obj[6] \* ih)

s = ymax - ymin

n = 1.0 \* ih/ s

if n < 5:#标记非机动车距离小于5米

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 0, 255), 2, 8)

m=1

if 5 < n :

cv.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 0, 0), 2, 8)

cv.putText(frame, str(obj[2]), (xmin, ymin), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1, 8)

print (j,m,i)

#针对行人优先级最高，非机动车其次，机动车最迟的优先级报警

if j == 1 and yanshi>100:

ser.write(b"3")

yanshi=0

if j==0 and m==1 and yanshi>100:

ser.write(b"2")

yanshi = 0

if j==0 and m==0 and i==1 and yanshi>100:

ser.write(b"1")

print('ok')

yanshi = 0

cv.putText(frame, "infer time(ms): %.3f, FPS: %.2f" % (inf\_end \* 1000, 1 / (inf\_end+float("1e-8"))), (10, 50),

cv.FONT\_HERSHEY\_SIMPLEX, 1.0, (255, 0, 255), 2, 8)

cv.namedWindow("Pedestrian Detection", 0)

cv.imshow("Pedestrian Detection", frame)

c = cv.waitKey(1)

if c == 27:

break

cv.waitKey(0)

cv.destroyAllWindows()

#播放音乐

def music():

ser = serial.Serial(port="COM9", baudrate=9600)#初始化串口，准备通信

ser.port = 'COM9'

ser.baudrate = 9600

ser.stopbits = 1

ser.bytesize = 8

ser.parity = 'N'

while (ser.isOpen() == 0):

print('打开串口失败')

ser.open()

while 1:#语音控制音乐的播放、切换与暂停

num=ser.readline().decode()

print(num)

if num=='9\r\n':

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\西安人的歌.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

if num=='8\r\n':

mixer.init() # 初始化混音器模块

mixer.music.load(r'C:\Users\Administrator\Desktop\消愁.mp3') # 载入待播放音乐文件

mixer.music.play() # 开始播放音乐流

if num == '7\r\n':

mixer.music.stop() # 结束音乐播放

if \_\_name\_\_ == "\_\_main\_\_":

ssd\_video\_demo()

import numpy as np

import cv2 as cv

import dlib

from scipy.spatial import distance

from PIL import Image, ImageDraw, ImageFont

from scipy.spatial import distance as dis

from imutils.video import FileVideoStream

from imutils.video import VideoStream

from imutils import face\_utils

import argparse

import imutils

import time

import math

from threading import Thread

# 调用人脸检测器

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

# 设定人眼标定点

LeftEye\_Start = 36

LeftEye\_End = 41

RightEye\_Start = 42

RightEye\_End = 47

Mouth\_Start = 48

Mouth\_End = 59

Radio=0.23  #横纵比阈值

#Radio=100

Low\_radio\_constant = 30  #意味着连续多少帧横纵比小于Radio小于阈值时，判断疲劳

Low\_radio\_constant\_mouth=15  #意味着连续多少帧横纵比小于Radio小于阈值时，判断疲劳

Mouth\_Moude\_Radio=1.3 #嘴部纵横比阈值

HAR\_THRESH=0.3 #头部姿态阈值

NOD\_AR\_CONSEC\_FRAMES=3 #帧数

hCOUNTER=0

# 世界坐标系(UVW)：填写3D参考点，该模型参考http://aifi.isr.uc.pt/Downloads/OpenGL/glAnthropometric3DModel.cpp

object\_pts = np.float32([[6.825897, 6.760612, 4.402142],  #33左眉左上角

                         [1.330353, 7.122144, 6.903745],  #29左眉右角

                         [-1.330353, 7.122144, 6.903745], #34右眉左角

                         [-6.825897, 6.760612, 4.402142], #38右眉右上角

                         [5.311432, 5.485328, 3.987654],  #13左眼左上角

                         [1.789930, 5.393625, 4.413414],  #17左眼右上角

                         [-1.789930, 5.393625, 4.413414], #25右眼左上角

                         [-5.311432, 5.485328, 3.987654], #21右眼右上角

                         [2.005628, 1.409845, 6.165652],  #55鼻子左上角

                         [-2.005628, 1.409845, 6.165652], #49鼻子右上角

                         [2.774015, -2.080775, 5.048531], #43嘴左上角

                         [-2.774015, -2.080775, 5.048531],#39嘴右上角

                         [0.000000, -3.116408, 6.097667], #45嘴中央下角

                         [0.000000, -7.415691, 4.070434]])#6下巴角

# 相机坐标系(XYZ)：添加相机内参

K = [6.5308391993466671e+002, 0.0, 3.1950000000000000e+002,

     0.0, 6.5308391993466671e+002, 2.3950000000000000e+002,

     0.0, 0.0, 1.0]# 等价于矩阵[fx, 0, cx; 0, fy, cy; 0, 0, 1]

# 图像中心坐标系(uv)：相机畸变参数[k1, k2, p1, p2, k3]

D = [7.0834633684407095e-002, 6.9140193737175351e-002, 0.0, 0.0, -1.3073460323689292e+000]

# 像素坐标系(xy)：填写凸轮的本征和畸变系数

cam\_matrix = np.array(K).reshape(3, 3).astype(np.float32)

dist\_coeffs = np.array(D).reshape(5, 1).astype(np.float32)

# 重新投影3D点的世界坐标轴以验证结果姿势

reprojectsrc = np.float32([[10.0, 10.0, 10.0],

                           [10.0, 10.0, -10.0],

                           [10.0, -10.0, -10.0],

                           [10.0, -10.0, 10.0],

                           [-10.0, 10.0, 10.0],

                           [-10.0, 10.0, -10.0],

                           [-10.0, -10.0, -10.0],

                           [-10.0, -10.0, 10.0]])

# 绘制正方体12轴

line\_pairs = [[0, 1], [1, 2], [2, 3], [3, 0],

              [4, 5], [5, 6], [6, 7], [7, 4],

              [0, 4], [1, 5], [2, 6], [3, 7]]

def get\_head\_pose(shape):# 头部姿态估计

    # （像素坐标集合）填写2D参考点，注释遵循https://ibug.doc.ic.ac.uk/resources/300-W/

    # 17左眉左上角/21左眉右角/22右眉左上角/26右眉右上角/36左眼左上角/39左眼右上角/42右眼左上角/

    # 45右眼右上角/31鼻子左上角/35鼻子右上角/48左上角/54嘴右上角/57嘴中央下角/8下巴角

    image\_pts = np.float32([shape[17], shape[21], shape[22], shape[26], shape[36],

                            shape[39], shape[42], shape[45], shape[31], shape[35],

                            shape[48], shape[54], shape[57], shape[8]])

    # solvePnP计算姿势——求解旋转和平移矩阵：

    # rotation\_vec表示旋转矩阵，translation\_vec表示平移矩阵，cam\_matrix与K矩阵对应，dist\_coeffs与D矩阵对应。

    \_, rotation\_vec, translation\_vec = cv.solvePnP(object\_pts, image\_pts, cam\_matrix, dist\_coeffs)

    # projectPoints重新投影误差：原2d点和重投影2d点的距离（输入3d点、相机内参、相机畸变、r、t，输出重投影2d点）

    reprojectdst, \_ = cv.projectPoints(reprojectsrc, rotation\_vec, translation\_vec, cam\_matrix,dist\_coeffs)

    reprojectdst = tuple(map(tuple, reprojectdst.reshape(8, 2)))# 以8行2列显示

    # 计算欧拉角calc euler angle

    # 参考https://docs.opencv.org/2.4/modules/calib3d/doc/camera\_calibration\_and\_3d\_reconstruction.html#decomposeprojectionmatrix

    rotation\_mat, \_ = cv.Rodrigues(rotation\_vec)#罗德里格斯公式（将旋转矩阵转换为旋转向量）

    pose\_mat = cv.hconcat((rotation\_mat, translation\_vec))# 水平拼接，vconcat垂直拼接

    # decomposeProjectionMatrix将投影矩阵分解为旋转矩阵和相机矩阵

    \_, \_, \_, \_, \_, \_, euler\_angle = cv.decomposeProjectionMatrix(pose\_mat)

    pitch, yaw, roll = [math.radians(\_) for \_ in euler\_angle]

    pitch = math.degrees(math.asin(math.sin(pitch)))

    roll = -math.degrees(math.asin(math.sin(roll)))

    yaw = math.degrees(math.asin(math.sin(yaw)))

    print('pitch:{}, yaw:{}, roll:{}'.format(pitch, yaw, roll))

    return reprojectdst, euler\_angle# 投影误差，欧拉角

def calculate\_Ratio(eye):

    """

    计算眼睛横纵比

    """

    d1 = distance.euclidean(eye[1], eye[5])

    d2 = distance.euclidean(eye[2], eye[4])

    d3 = distance.euclidean(eye[0], eye[3])

    ratio = (d1 + d2) / (2 \* d3)

    return ratio

def mouthRatio(mouth):

    """

    计算嘴巴横纵比

    """

    left=dis.euclidean(mouth[2],mouth[10])

    mid=dis.euclidean(mouth[3],mouth[9])

    right=dis.euclidean(mouth[4],mouth[8])

    horizontal=dis.euclidean(mouth[0],mouth[6])

    return 10.0\*horizontal/(3.0\*left+4.0\*mid+3.0\*right)

def main():

    """

    主函数

    """

    alarm = False  # 初始化眼睛警报

    mouth\_alarm=False #初始化嘴巴警报

    frame\_counter=0  # 连续帧计数

    frame\_counter\_mouth=0

    mouth=0

    eye=0

    head=0

    hCOUNTER=0

    hTOTAL=0

    cap = cv.VideoCapture(0)  # 0摄像头摄像

    while cap.isOpened():

        ret, frame = cap.read()  # 读取每一帧

        frame = cv.flip(frame, 1)

        if ret:

            gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY)

            rects = detector(gray, 0)  # 人脸检测

            for rect in rects:

                shape = predictor(gray, rect)

                points = np.zeros((68, 2), dtype=int)

                for i in range(68):

                    points[i] = (shape.part(i).x, shape.part(i).y)

                # 获取眼睛特征点

                Lefteye = points[LeftEye\_Start: LeftEye\_End + 1]

                Righteye = points[RightEye\_Start: RightEye\_End + 1]

                Mouth=points[Mouth\_Start: Mouth\_End + 1]

                # 计算眼睛横纵比

                Lefteye\_Ratio = calculate\_Ratio(Lefteye)

                Righteye\_Ratio = calculate\_Ratio(Righteye)

                mean\_Ratio = (Lefteye\_Ratio + Righteye\_Ratio)/2  # 计算两眼平均比例

                #计算嘴部横纵比

                Mouth\_Ratio=mouthRatio(Mouth)

                #print(Mouth\_Ratio)

                # 计算凸包

                left\_eye\_hull = cv.convexHull(Lefteye)

                right\_eye\_hull = cv.convexHull(Righteye)

                mouth\_hull=cv.convexHull(Mouth)

                # 绘制轮廓

                cv.drawContours(frame, [left\_eye\_hull], -1, [0, 255, 0], 1)

                cv.drawContours(frame, [right\_eye\_hull], -1, [0, 255, 0], 1)

                cv.drawContours(frame, [mouth\_hull], -1, [0, 255, 0], 1)

                # 眨眼判断

                if mean\_Ratio<Radio:

                    frame\_counter+=1

                    #print("困了")

                    #print(frame\_counter)

                    if frame\_counter>=Low\_radio\_constant:

                        # 发出警报

                       # print("闭眼")

                        eye=1

                        if not alarm:

                            alarm = True

                        cv.putText(frame, "eye closing", (40,70),

                                   cv.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

                else:

                    #print(123456)

                    eye=0

                    alarm=False

                    frame\_counter=0

                #打哈欠判断

                if Mouth\_Ratio<Mouth\_Moude\_Radio:

                    frame\_counter\_mouth+=1

                    if frame\_counter\_mouth>=Low\_radio\_constant\_mouth:

                        # 发出警报

                        mouth=1

                        if not mouth\_alarm:

                            mouth\_alarm = True

                        cv.putText(frame, "yawning", (40, 100),

                                   cv.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

                else:

                    mouth=0

                    mouth\_alarm = False

                    frame\_counter\_mouth=0

                #头部姿态

                shape = face\_utils.shape\_to\_np(shape)

                reprojectdst, euler\_angle = get\_head\_pose(shape)

                har = euler\_angle[0, 0]# 取pitch旋转角度

                if har > HAR\_THRESH:    # 点头阈值0.3

                    hCOUNTER+=1

                else:

                # 如果连续3次都小于阈值，则表示瞌睡点头一次

                    if hCOUNTER >= NOD\_AR\_CONSEC\_FRAMES:# 阈值：3

                       hTOTAL += 1

            # 重置点头帧计数器

                    hCOUNTER = 0

                if  hTOTAL==3:

                    hTOTAL=0

                    head=1

                else:

                    head=0

                for start, end in line\_pairs:

                    cv.line(frame, reprojectdst[start], reprojectdst[end], (0, 0, 255))

                for (x, y) in shape:

                    cv.circle(frame, (x, y), 1, (0, 0, 255), -1)

                # 显示结果

                if mouth==1 and eye==1:

                    cv.putText(frame, "serious", (40, 40),

                                   cv.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

                elif mouth==1 or eye==1 or head==1:

                    cv.putText(frame, "middle", (40, 40),

                                   cv.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

                else:

                    cv.putText(frame, "normal", (40, 40),

                                   cv.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

                cv.putText(frame, "Eye Ratio{:.2f}".format(mean\_Ratio), (200, 70),

                           cv.FONT\_HERSHEY\_SIMPLEX, 0.7, [0, 0, 255], 2)

                cv.putText(frame, "Mouth Ratio{:.2f}".format(Mouth\_Ratio), (200, 40),

                           cv.FONT\_HERSHEY\_SIMPLEX, 0.7, [0, 0, 255], 2)

                cv.putText(frame, "HEAD Ratio{:.2f}".format(hTOTAL), (200, 100),

                           cv.FONT\_HERSHEY\_SIMPLEX, 0.7, [0, 0, 255], 2)

            #界面显示

            cv.imshow("test", frame)

            if cv.waitKey(1) & 0xFF == ord('q'):

                break

    cap.release()

    cv.destroyAllWindows()

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]

(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]

(mStart, mEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["mouth"]

if \_\_name\_\_ == '\_\_main\_\_':

    hCOUNTER=0

    main()

from unittest import result

from aip import AipBodyAnalysis

import json

import cv2

from numpy import var

from sqlalchemy import true

from threading import Thread

""" 你的 APPID AK SK """

#可修改的数据

""" 你的 APPID AK SK """

APP\_ID = '26752080'

API\_KEY = 'YcYCYLLggUMRyuegdbKGjl5S'

SECRET\_KEY ='nPUE7eWx4jdixjwMluR6mkqdwxeRQWFE'

""" 图片地址 """

image\_a="diver.jpeg"

capture = cv2.VideoCapture(0)#0为默认摄像头

def camera():

    while True:

        # 获得图片

        ret, frame = capture.read()

        #cv2.imshow("窗口名称", "窗口显示的图像)

        # 显示图片

        #cv2.putText(frame, "yawning", (40, 100),

                                   #cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

        cv2.imshow('frame',frame)

        if cv2.waitKey(1) == ord('q'):

            break

Thread(target=camera).start()  # 引入线程防止在识别的时候卡死

""" 需要的参数（0则为需要，1则为不需要） """

smoke=0 #吸烟

cellphone=0 #打手机

not\_buckling\_up=1 #未系安全带

both\_hands\_leaving\_wheel=1 #双手离开方向盘

not\_facing\_front=1 #视角未看前方

""" 可按实际情况修改 """

""" "1"，表示左舵车 "0"，表示右舵车 """

wheel\_location=1

client=AipBodyAnalysis(APP\_ID, API\_KEY, SECRET\_KEY) #生成key

number=1

while True:

    video = cv2.VideoCapture(0)

    ret,frame = capture.read()

    cv2.imshow('frame', frame)

    #image=get\_file\_content(frame)

    image = cv2.imencode('.jpg', frame)[1]

    """ 调用驾驶行为分析 """

    client.driverBehavior(image)

    """ 如果有可选参数 """

    options = {}

    options\_str=""

    if smoke==0 : options\_str = options\_str + "smoke,"

    if cellphone==0 : options\_str = options\_str + "cellphone,"

    if not\_buckling\_up==0 : options\_str = options\_str + "not\_buckling\_up,"

    if both\_hands\_leaving\_wheel==0 : options\_str = options\_str + "both\_hands\_leaving\_wheel,"

    if not\_facing\_front==0 : options\_str = options\_str + "not\_facing\_front,"

    options["type"] = options\_str

    options["wheel\_location"] = wheel\_location

    #print("\*\*\*\*\*\*\*\*\*\*\*\*\*")

    #print(options)

    """ 带参数调用驾驶行为分析 """

    a=client.driverBehavior(image, options)

    #print(json.dumps(a,indent=3)) #格式化输出

    data=json.dumps(a)

    data\_dict=json.loads(data)

    print(number)

    print(data\_dict)

    result=json.loads()

    print("\*\*\*\*\*\*\*\*\*\*\*\*\*")

    data2=data\_dict["person\_info"]

    data3=data2[0]

    data4=data3['attributes']

    data5=data4['smoke']

    smoke\_score=data5['score']

    data6=data4['cellphone']

    phone\_score=data6['score']

    if(smoke\_score>0.48):

        print("正在抽烟")

    if(phone\_score>0.69):

        print("正在打电话")

    print("\*\*\*\*\*\*\*\*\*\*\*\*\*")

    if cv2.waitKey(1) & 0xFF == ord('q'):

             break

video.release()

cv2.destroyAllWindows()