目录

[2、前端组件模块 5](#_Toc13497061)

[2.1 程序描述 5](#_Toc13497062)

[2.2 功能 5](#_Toc13497063)

[2.3 流程逻辑 6](#_Toc13497064)

[.2.4 输入项 7](#_Toc13497065)

[2.5 输出项 7](#_Toc13497066)

[2.6 设计方法 8](#_Toc13497067)

[2.7 接口 8](#_Toc13497068)

[3物体检测模块 8](#_Toc13497069)

[3.1 程序描述 8](#_Toc13497070)

[3.2 功能 9](#_Toc13497071)

[3.3 性能要求 9](#_Toc13497072)

[3.4 输入项 9](#_Toc13497073)

[3.5 输出项 10](#_Toc13497074)

[3.6 设计方法 10](#_Toc13497075)

[3.8 接口 13](#_Toc13497076)

[3.9 存储分配 16](#_Toc13497077)

[3.10 注释设计 16](#_Toc13497078)

[3.11 限制条件 16](#_Toc13497079)

[3.12 测试计划 16](#_Toc13497080)

[4语义分割模块 17](#_Toc13497081)

[4.1程序描述 17](#_Toc13497082)

[4.2功能 17](#_Toc13497083)

[4.3性能 18](#_Toc13497084)

[4.4输入项 18](#_Toc13497085)

[4.5输出项 18](#_Toc13497086)

[4.6设计方法（算法） 18](#_Toc13497087)

[4.6.1 前端模块设计 18](#_Toc13497088)

[4.6.2 后端模块设计 19](#_Toc13497089)

[4.6.3 模型模块设计 19](#_Toc13497090)

[4.7 流程逻辑 20](#_Toc13497091)

[4.8 接口 21](#_Toc13497092)

[4.9存储分配 21](#_Toc13497093)

[4.10 注释设计 21](#_Toc13497094)

[4.11 限制条件 22](#_Toc13497095)

[4.12 测试计划 22](#_Toc13497096)

[4.13 尚未解决的问题 22](#_Toc13497097)

[5交通灯判别模块 23](#_Toc13497098)

[5.1 程序描述 23](#_Toc13497099)

[5.2功能 23](#_Toc13497100)

[5.3性能 24](#_Toc13497101)

[5.4输入项 24](#_Toc13497102)

[5.5输出项 24](#_Toc13497103)

[5.6设计方法（算法） 24](#_Toc13497104)

[5.7流程逻辑 24](#_Toc13497105)

[5.8接口 24](#_Toc13497106)

[5.9存储分配 25](#_Toc13497107)

[5.10注释设计 25](#_Toc13497108)

[5.11限制条件 25](#_Toc13497109)

[5.12测试计划 25](#_Toc13497110)

[5.13尚未解决的问题 25](#_Toc13497111)

[6路径规划模块 26](#_Toc13497112)

[6.1程序描述 26](#_Toc13497113)

[6.2功能 26](#_Toc13497114)

[6.3性能 27](#_Toc13497115)

[6.4输入项 27](#_Toc13497116)

[6.5输出项 27](#_Toc13497117)

[6.6 UI设计说明 28](#_Toc13497118)

[6.7 流程逻辑 28](#_Toc13497119)

[6.8 接口 28](#_Toc13497120)

[6.9存储分配 28](#_Toc13497121)

[6.10 注释设计 28](#_Toc13497122)

[6.11 限制条件 30](#_Toc13497123)

[6.12 测试计划 30](#_Toc13497124)

[6.13 尚未解决的问题 30](#_Toc13497125)

[7.日志分析模块 31](#_Toc13497126)

[7.1程序描述 31](#_Toc13497127)

[7.2 功能 31](#_Toc13497128)

[7.3 性能 32](#_Toc13497129)

[7.4 输入项 32](#_Toc13497130)

[7.5 输出项 32](#_Toc13497131)

[7.6 UI设计说明 32](#_Toc13497132)

[7.7 流程逻辑 32](#_Toc13497133)

[7.8 接口 33](#_Toc13497134)

[7.9 存储分配 33](#_Toc13497135)

[7.10 注释设计 33](#_Toc13497136)

[7.11 限制条件 33](#_Toc13497137)

[7.12测试计划 33](#_Toc13497138)

[7.13 尚未解决的问题 33](#_Toc13497139)

[8.即时传输模块 34](#_Toc13497140)

[8.1程序描述 34](#_Toc13497141)

[8.2 功能 34](#_Toc13497142)

[8.3 性能 34](#_Toc13497143)

[8.4 输入项 34](#_Toc13497144)

[8.5 输出项 34](#_Toc13497145)

[8.6 UI设计说明 34](#_Toc13497146)

[8.7 流程逻辑 34](#_Toc13497147)

[8.8 接口 35](#_Toc13497148)

[8.9 存储分配 35](#_Toc13497149)

[8.10 注释设计 35](#_Toc13497150)

[8.11 限制条件 35](#_Toc13497151)

[8.12测试计划 35](#_Toc13497152)

[8.13 尚未解决的问题 35](#_Toc13497153)

# 2、前端组件模块

## 2.1 程序描述

前端组件模块负责搭建一个可供展示的网页界面，该界面展示经过模型算法处理后的车载摄像头数据、毫米波雷达数据和实时构建车辆行径地图。用户可以根据网页显示实时监控车辆前方道路状态、车辆周围状态、车辆已行驶地图。

## 2.2 功能

前端组件模块下面包括八个小模块：

1. 头部导航栏模块：用户在此处选取需要展示的模块，有主页（一起展示）、图像检测、毫米波检测和地图构建，并且用户可以选择进行实时监控或者运行历史数据。
2. 左侧边栏：此模块属于图像模块的二级菜单，供用户选择视频流需要选择展示的功能，有物体检测，语义分割和交通灯判别。
3. 右侧边栏：此模块是对展示功能的数值描述，如当前进度，当前准确率，各类别准确率，历史数据文件选择。并且在前方存在危险的情况下进行只能提示。
4. 物体检测功能模块：负责在页面中间的车载视频展示中加入物体检测功能作用后的结果。
5. 语义分割功能模块：负责在页面中间的车载视频展示中加入语义分割功能作用后的结果。
6. 交通灯判定模块：负责在页面中间的车载视频展示中加入交通灯判定功能作用后的结果。
7. 毫米波功能模块：负责在页面中间的车辆俯视图中展示车辆周围信息。
8. 地图规划功能模块：负责在页面中间的空白处构建出车辆行驶路径。

## 2.3 流程逻辑

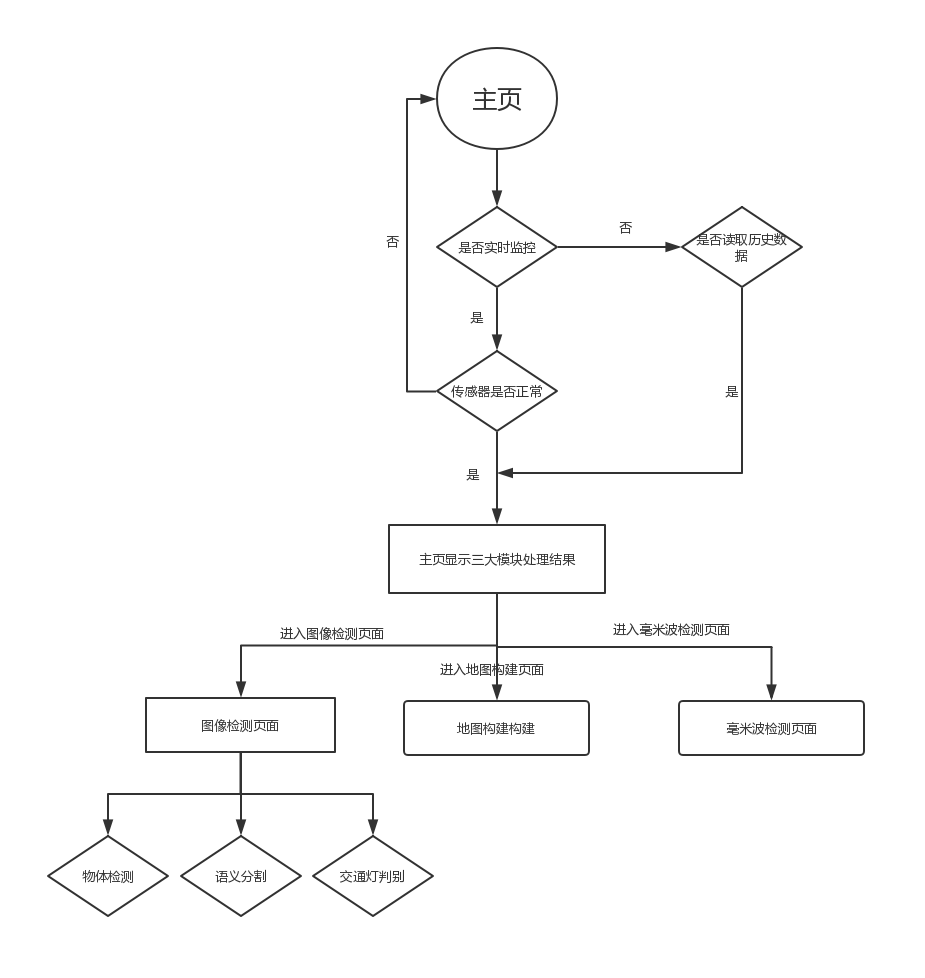


图2-1 网页操作流程

当用户打开网页终端后，操作流程如下：

1. 输入地址后，主页中选择实时监控或者是运行后台中存储好的数据。
2. 选择实时监控之后，会提示等待判断传感器是否连接，之后在主页中同时显示图像检测、毫米波检测和地图构建结果。
3. 选择运行历史数据后，提示选择具体的数据，之后等待数据加载，之后在主页中同时显示图像检测、毫米波检测和地图构建结果。
4. 在主页中可以在头部导航栏模块选择单独展示一个模块的功能。
5. 运行结束，结果保存，提示用户。

## .2.4 输入项

（1）：经过物体检测功能处理后的视频流，当前检测结果。

（2）：经过语义分割功能处理后的视频流，当前检测结果。

（3）：经过交通灯判断功能处理后的视频流，当前检测结果。

（4）：经过算法处理后的毫米波传感器信号。

（5）：构建地图需要的相关信息。

## 2.5 输出项

图像检测模块输出物体检测、语义分割、交通灯检测的结果。

（1）：针对物体检测，对每一个检测出的物体使用方框进行标记，针对不同的种类使用不同颜色进行区别，并且在右侧边栏对检测物体种类、置信度、对危险物体进行红色加强提醒。如果处于运行历史数据状态，则显示当前进度、当前准确率。

地图构建模块会在网页中实时构建车辆行驶过的路径，并且标记出车辆的当前位置。

毫米波雷达模块会对车辆周围的物体进行感应，最终在网页中的毫米波俯视图中标记出来，并且在右侧边栏上对危险物体进行智能提示。

## 2.6 设计方法

采用MVVM设计模式，利用前端框架Angular和CSS模板Semantic-UI进行构建，采用数据-视图分离。Service.ts是针对各个模块提供数据服务的，如leftbar.service.ts和rightbar.service.ts。用户通过访问service对象获取展示数据和实时显示。

## 2.7 接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数名 | 形参说明 | 返回值 | 函数功能 |
| 1 | isImageDetection() | 无 | 布尔值，返回当前页面状态 | 检查当前网页窗口是否在图像检测页面，如果是，则返回真，否则返回假。 |
| 2 | isOD() | 无 | 布尔值，返回当前页面状态 | 检测是否选择了物体检测功能，如果是则返回真，否则则返回假 |
| 3 | getODactivate() | 无 | 布尔值，返回物体检测功能状态 | 获取当前物体检测功能是否激活状态。 |
| 4 | showSS() | 无 | 无 | 打开语义分割功能，打开语义分割右侧边栏。 |
| 5 | getSSactivate() | 无 | 布尔值，获取语义分割功能状态 | 获取当前语义分割功能的状态。属于激活状态则为真，否则为假。 |
| 6 | showTL() | 无 | 无 | 打开交通灯检测功能；打开交通灯右侧边栏。 |
| 7 | getTLactivate() | 无 | 布尔值，返回交通灯检测功能当前状态。 | 获取交通灯检测功能当前状态。如果正常则为真，否则为假。 |

# 3物体检测模块

## 3.1 程序描述

物体检测模块分为前端的视频流显示部分、后端深度网络模型和网站后端部分。当用户在导航栏选择了图像检测并且在左侧边栏选择了物体检测功能之后，前端部分发送给后端请求物体检测数据，后端接受到请求后调用模型进行物体检测，后端收集好数据进行回传给前端。

## 3.2 功能

物体检测模块主要对视频流上的物体进行类型判断和位置识别，并且在视频中标注出其具体类别和物理位置。通过右侧边栏的表单对处理结果进行文字描述。

物体检测由三个子模块组成：  
 （1）：前端显示模块，前端显示模块分为视频显示框和右侧边栏信息框，视频显示框针对摄像头传感器采集到的视频数据进行分析，最终显示经过物体检测功能处理后的视频图像；右侧边栏对检测到的物体进行文字描述，如物体类别，物体位置等；对危险距离的物体采用突出提示。

（2）：网站后端模块：针对前端提出的调用物体检测功能或者停用物体检测功能进行接受，并且操作后端深度网络模型进行相应的操作。

（3）：后端深度网络模型：算法层面的模型，针对请求调用相应的网络进行运算。

## 3.3 性能要求

对实时视频流的检测不低于30帧每秒，对历史数据视频流（数据集）的检测，不低于40帧每秒，且准确率高于78%；保证处于危险位置的物体可以完全检测出；在昏暗环境下检测物体具有较强的适应性。

## 3.4 输入项

车载摄像头采集的视频流信息，最终转化为数帧图片进行输入。

## 3.5 输出项

通过物体检测算法后的结果标记数据。

## 3.6 设计方法

在物体检测模块加入到前端模块进行实时检测之前，需要对其算法网络模型进行训练，数据集使用到的JPEG格式文件需要通过trans\_array函数进行转换为numpy格式的三维宽高相等的数字型矩阵（3x224x224）;算法模型类class Network通过torch框架进行构建，通过函数\_\_init\_\_对其拥有的每一层网络进行声明，这些网络层是通过conv2D、BatchNorm2D、ReLU、MaxPool（）等基础网络结构进行复合而得到的。模型参数存在于网络结构中，使用浮点矩阵dtype=float32进行存放，函数forward()使之前声明好的网络层有序地连接起来，构成一整张网络；通过train（）函数对构建成功的网络进行训练，在训练期间需要通过CrossEntropyLoss()定义的损失函数进行backward()反向传播从而更新参数。

（2）前端请求包

当前端请求执行物体检测任务的时候，需要向后端传递的请求包应该包括以下几点：运行数据名称run\_data，当此字段为’realtime’时，检查传感器是否正常工作，非正常工作则返回报错包，当此字段为其他字符串时，检查数据库是否存在数据，若无此数据则返回报错包。

（3）模型输出数据

当Numpy格式的图片矩阵传入算法模型，在图像进行预测之后，对模型输出进行读取和实时传输阶段，定义数据结构OD\_data进行存储数据，将所有类别进行编号，并且将预测的类别信息转换成编号放入成员category中,并且将置信度估计放入其成员变量prob中。一张图片中可能存在多个物体，所以一张图片中会生成多个OD\_data。每当生成一张图片的全部预测之后，后端将OD\_data数组上的信息绘制到原始图片。

（4）后端返回数据包

后端针对已经生成的OD\_data数组进行格式化，连同其他功能的模块一起发送给前端。

3.7 程序流程图

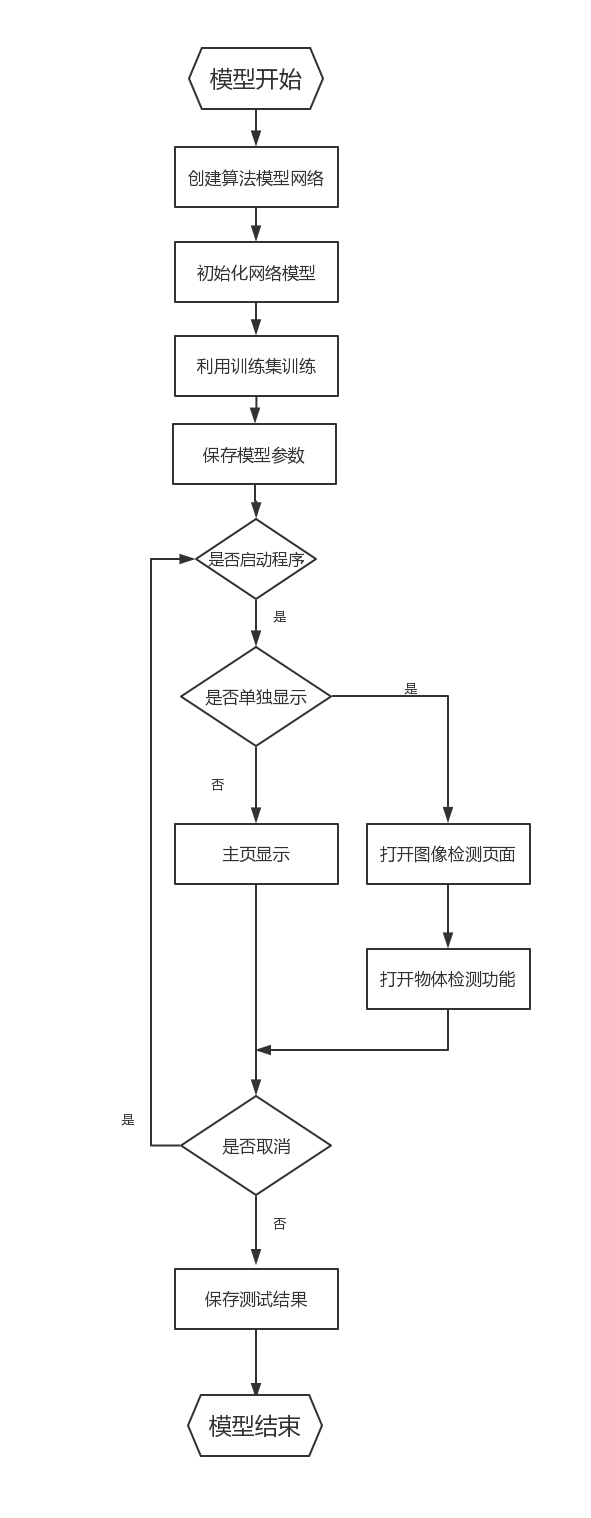


图3-1 物体检测模块流程图

## 3.8 接口

该模块接口分为三个部分，前端接口、算法模型接口，后端传输接口，前端通过show\_OD函数调用后端启动物体检测算法，后端通过collect\_OD\_data进行对模型结果的收集，其中还涉及到了格式转换函数trans\_array，存储函数save\_OD\_params、save\_OD\_data等，具体函数接口如表3-1所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数名 | 形参说明 | 返回值 | 函数功能 |
| 1 | ShowOD(any:status,$event) | Status表明当前网页所属状态，event会记录鼠标或者键盘的所有动作。 | 布尔值，函数是否成功 | 调用隐藏函数hide\_others()对其他功能板块进行隐藏，并显示物体检测页面，调用send\_OD\_request函数。调用check\_sensor函数。 |
| 2 | Hide\_others(any:status) | Status表明当前网页所属状态。 | 布尔值，函数是否成功 | 隐藏其他功能的前端模板。 |
| 3 | Show\_OD\_rightbar  (data\_source) | Data\_source表明数据源是属于历史数据还是实时检测 | 布尔值，函数是否成功 | 显示右侧边栏的信息。 |
| 4 | Check\_sensor(string:cerma) | 字符型数据，表示需要检查的传感器类型 | 成功或者是具体的错误代码。 | 检查传感器是否正常工作。 |
| 5 | Trans\_array(file\_name) | File\_name表明需要进行转换的图片文件，一般为JPEG格式 | 返回numpy.float32类型的3维矩阵 | 将JPEG格式的图片转换为数字矩阵 |
| 6 | Collect\_OD\_data(array:out) | out是算法模型的输出1x1004的数组。 | 返回剔除无用信息后的1x2数组 | 将模型的输出规范化。 |
| 7 | Save\_OD\_params(net) | Net是网络模型类 | 布尔值，是否成功 | 将模型结构、参数持久化。 |
| 8 | Save\_OD\_data(array:1x2) | 规范化后的数组 | 布尔值，是否成功 | 将规范化的输出进行存储。 |
| 9 | Start\_OD\_model(data\_loader) | Data\_loader表明分析数据是属于传感器传输还是历史数据 | 布尔值，OD算法模块是否成功启动。 | 启动后端的物体检查算法模块。 |
| 10 | Net.forward(array) | 三维numpy数组，通过图像转换得到 | Out模型预测结果 | 模型进行前向传播得到预测结果。 |
| 11 | Net.backward() | 类内函数 | 无 | 对模型的训练参数进行反向传播更新。 |
| 12 | Cross\_entropy\_loss(out, real) | Out表示模型输出，real表示真实输出 | 输出out和real的交叉熵 | 对预测输出和真实输出进行比较得到交叉熵。 |

## 3.9 存储分配

传感器实时数据存储:/data/camera

历史数据存储:/data\_set/camera/train 和/data\_set/camera/val

## 3.10 注释设计

对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

对变量的注释：在该模块中程序用到的主要变量都作出必要的说明。

对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明。

对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性

## 3.11 限制条件

需要显卡内存大于2G，并且项目前端成功启动。

## 3.12 测试计划

测试平台：

操作系统：Ubuntu16.04 LST

前端框架：Angular8.0.4 flask1.0

# 4语义分割模块

## 4.1程序描述

语义分割是计算机视觉中的基本任务，在语义分割中我们需要将视觉输入分为不同的语义可解释类别，「语义的可解释性」即分类类别在真实世界中是有意义的。例如，我们可能需要区分图像中属于汽车的所有像素，并把这些像素涂成蓝色。而本模块的任务就是对摄像头采集到的视频进行实时分割，并把处理后的视频实时传输到前端进行可视化展示。

## 4.2功能

本模块由以下三个子模块组成:

（1）前端可视化模块

前端作为可视化展示页面，其输入为处理后的视频流数据，其输出是实时的视频流显示以及视频流的相关详细信息（视频流当中各个颜色对应的类别、准确率等）。其效果如图2-1所示。



图2-1 前端效果图

（2）后端逻辑控制模块

后端作为整个语义分割模块的中枢神经，起到承上启下的作用，控制着传感器（摄像头）与模型模块的通信以及前端的数据传输。

（3）模型模块

模型模块接受传感器实时传输来的原始视频流，经过相关处理，再通过后端实时传输给前端进行可视化展示。

## 4.3性能

对与传感器采集到的视频流数据能够达到实时的分割并且传输到前端可视化界面进行展示，效果能够达到120FPS/S，且准确率达到68%左右。

## 4.4输入项

输入项分为以下两个不同内容：

（1）当网页选择的是实时处理的功能的时候，此时整个模块的输入就是传感器（摄像头）实时采集到的原始视频流；

（2）当网页选择的是处理历史数据集的功能的时候，此时整个模块的输入就是本地文件夹当中保存的历史数据集。

## 4.5输出项

输出项分为以下两个不同内容：

（1）当网页选择的是实时处理的功能的时候，此时整个模块的输出就是模型模块实时处理后的视频流以及视频流当中每个颜色对应的物体种类。

（2）当网页选择的是处理历史数据集的功能的时候，此时整个模块的输出就是历史数据集的处理结果、预测准确率以及每个颜色对应的物体种类。

## 4.6设计方法（算法）

### 4.6.1 前端模块设计

前端模块采用的技术为Angular+Semantic UI。

AngularJS诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：MVC、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。

Semantic UI 是一款很好用的CSS工具，事实上，你可以认为它是一款不错的化妆品，因为它比普通的CSS制作的界面更好看一些，也更容易。最大的特点就像是它的名字一样，你可以通过字面的语意来制作你的网页，如果你想要一个按钮，就直接写ui button, 你想让它变成红色，只需要写ui red button，不需要知道红色其实是#B03060。所以，我们只要描述出来我们想要的网页，sementic UI 就会将这个网页展示出来。我们用来描述的语言，在这里被叫做形容词，例如：segement,item, very large, fixed menu,inverted等等。

### 4.6.2 后端模块设计

后端模块使用的编程语言是python，其基本框架是Flask。

Flask是一个轻量级的可定制框架，使用Python语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合MVC模式进行开发，开发人员分工合作，小型团队在短时间内就可以完成功能丰富的中小型网站或Web服务的实现。另外，Flask还有很强的定制性，用户可以根据自己的需求来添加相应的功能，在保持核心功能简单的同时实现功能的丰富与扩展，其强大的插件库可以让用户实现个性化的网站定制，开发出功能强大的网站。其工作流程图如2-2所示 。

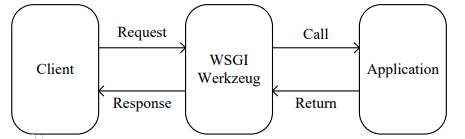


图2-2 Flask工作流程图

### 4.6.3 模型模块设计

本模块的输入是视频流当中的每一帧图像，整个模型网络结果如图2-3所示，分为以下几个部分：

（1）下采样学习模块：在该模块中，使用了三个层。这3个层确保了低层次特征能被有效地共享和使用。第一层是一个标准的卷积层(Conv2D)，其它两个层则为 depthwise separable 卷积层(DSConv)。这三个层都是使用的步长为2，并且后面跟着一个BN层和Relu层。

（2）全局特征提取模块：全局特征提取器目的在于捕捉图像分割的全局环境信息。和常用的 two-branch 方法不同，它是操作在图像的低分辨率输入上，我们的模块则直接将 learn to downsample 模块的输出作为输入（约为原始输入的1818\frac{1}{8}81大小），模块的详细结构如表1所述。我们使用了 MobileNet-V2 中高效的 bottleneck residual 模块（表2）。当输入和输出的大小一样时，我们在 bottleneck residual blocks 上使用 residual connection。Bottleneck block 使用了一个高效的 depthwise separable 卷积，使得参数个数和浮点数计算更少。在末端增加了一个金字塔池化模块来聚合不同区域的环境信息。

（3）特征融合模块：通过add操作，使两个分支的特征叠加起来使用。

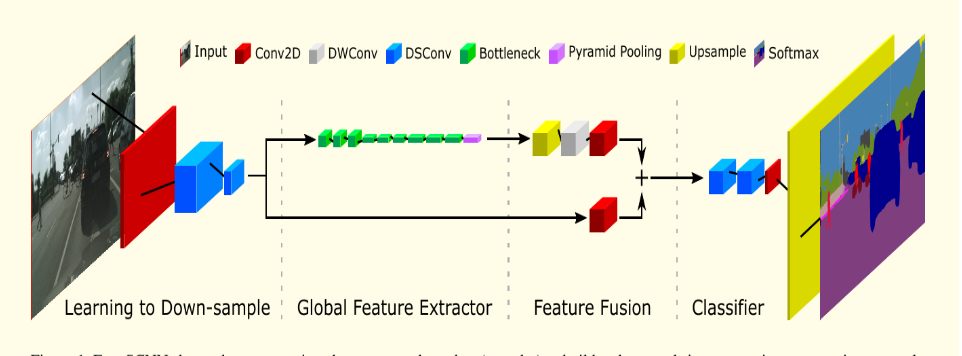


图2-3 模型网络结构图

## 4.7 流程逻辑

当用户通过浏览器进入系统网页的时候，首选需要选择启动语义分割功能以及处理的数据集来源。如果选择数据集来源于传感器的实时传输，那么就会通过后端进行传感器和模型之间的数据集实时传输，并且把处理后的数据实时传输到前端页面进行可视化展示，由于数据集来源于传感器的实时传输，所以在前端页面只会展示视频的处理结果以及每个颜色对应的类别。如果选择的是本地的历史数据集，那么最终的展示结果会增加一个预测准确率等评测信息。其流程如2-4所示。



图2-4 模块流程图

## 4.8 接口

## 4.9存储分配

## 4.10 注释设计

（1）对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

（2）对变量的注释：在该模块中程序用到的主要变量都做出必要的说明。

（3）对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明。

（4）对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性。

## 4.11 限制条件

## 4.12 测试计划

## 4.13 尚未解决的问题

# 5交通灯判别模块

## 5.1 程序描述

交通灯检测模块分为前端的视频流显示部分、后端深度网络模型和网站后端部分。当用户在导航栏选择了图像检测并且在左侧边栏选择了交通灯检测功能之后，前端部分发送给后端请求红绿灯检测数据，后端接受到请求后调用模型进行红绿灯检测，后端收集好数据进行回传给前端。

## 5.2功能

交通灯检测模块主要对视频流上的交通灯进行类型判断和位置识别，并且在视频中标注出其具体类别和物理位置，并通过右侧边栏对处理结果进行文字描述。

物体检测由三个子模块组成：  
 （1）前端显示模块，前端显示模块分为视频显示框和右侧边栏信息框，视频显示框针对摄像头传感器采集到的视频数据进行分析，最终显示经过红绿灯检测功能处理后的视频图像；右侧边栏对检测到的红绿灯进行文字描述，如红绿灯类别，位置等。

（2）网站后端模块：针对前端提出的调用红绿灯检测功能或者停用红绿灯检测功能进行接受，并且操作后端深度网络模型进行相应的操作。

（3）后端深度网络模型：算法层面的模型，针对请求调用相应的网络进行运算。

## 5.3性能

对与传感器采集到的视频流数据能够进行红绿灯检测并且传输到前端可视化界面进行展示，保证红绿灯的判断准确。

## 5.4输入项

车载摄像头采集的视频流信息，最终转化为数帧图片进行输入。

## 5.5输出项

通过红绿灯检测算法后的结果标记数据。

## 5.6设计方法（算法）

待

## 5.7流程逻辑

用户通过进入系统，选择启动红绿灯检测功能，通过后端进行传感器和模型之间的数据实时传输，在系统内进行红绿灯检测，并且把处理后的数据实时传输到前端页面进行可视化展示，前端页面展示视频的处理结果-即红绿灯的相关信息。

## 5.8接口

本模块接口分为三个部分，前端接口、算法模型接口、后端传输接口。

## 5.9存储分配

## 5.10注释设计

对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

对变量的注释：在该模块中程序用到的主要变量都作出必要的说明。

对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明。

对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性。

## 5.11限制条件

## 5.12测试计划

## 5.13尚未解决的问题

# 6路径规划模块

## 6.1程序描述

路径规划模块完成对传感器数据、各模块结果的整合展示，并以车辆部署图结合地图的形式展示。

## 6.2功能

本模块由以下三个子模块组成:

（1）视频可视化模块

前端作为可视化展示页面，其输入为处理后的视频流数据，其输出是实时的视频流显示以及视频流的相关详细信息（视频流当中各个颜色对应的类别、准确率等）。其效果如图2-1所示。



图2-1 前端效果图

（2）传感器数据展示

在车模型在标注传感器数据，如IMU、GPS定位数据，毫米波感知物体距离。

（3）SLAM地图模块

模型模块接受传感器实时传输来的原始视频流，经过相关处理，再通过后端实时传输给前端进行可视化展示。已

## 6.3性能

对与传感器采集到的视频流数据能够达到实时的分割并且传输到前端可视化界面进行展示，效果能够达到120FPS/S，且准确率达到68%左右。

## 6.4输入项

ROS传输的各传感器数据

## 6.5输出项

输出项分为以下两个不同内容：

（1）当网页选择的是实时处理的功能的时候，此时整个模块的输出就是模型模块实时处理后的视频流和对应构建的地图。

（2）当网页选择的是处理历史数据集的功能的时候，此时整个模块的输出就是历史数据集的处理结果和完成的地图。

## 6.6 UI设计说明



在主页面中展示SLAM过程和地图，在信息栏中展示传感器数据。

## 6.7 流程逻辑

6.8 接口

6.9存储分配

6.10 注释设计

（1）对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

（2）对变量的注释：在该模块中程序用到的主要变量都做出必要的说明。

（3）对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明。

（4）对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性。

6.11 限制条件

6.12 测试计划

6.13 尚未解决的问题

# 7.日志分析模块

## 7.1程序描述

本模块以界面的形式向用户提供日志文件的管理。通过本模块用户可以通过日志文件观察系统的历史操作和运行情况。

本模块的主要涉及思想是：以Web界面方式提供给用户操作系统界面和接受用户查询及用户自定义规则设置。采用B/S框架。

查询管理模块由显示模块、业务模块、业务处理模块等三个子模块组成。其中：

（1）显示模块用于向客户端用户提供一系列的页面视图，接受客户端的访问，为客户端请求返回查询内容，以及接受客户端输入的数据，并交付给服务器进行处理。

（2）业务模块负责将显示模块中涉及的业务处理提交给相应的业务处理模块，处理完成后，将处理后的结果交给显示模块，其作用相当于一个控制器。

（3）业务处理模块对业务中涉及到的请求和数据进行处理并进行持久性存储。

## 7.2 功能

日志分析接口模块主要由以下几个模块组成：

（1）规则管理模块

本模块为用户提供规则文件查看，以方便用户对当前的使用规则进行了解；提供日志文件修改，以方便用户对当前规则文件进行查询、修改、添加和删除等操作。

（2）日志查询

本模块用户输入要查询的时间端或者要查询的种类，能够对历史日志信息进行筛选显示。

（3）日志分析

本模块用户输入要统计的日志信息，根据该信息用户可以查看历史日志的统计数据。

## 7.3 性能

结果集数目＜100，响应时间＜1秒，结果集数目＞10，使用分页方式显示。

## 7.4 输入项

（1）系统在每个时间段的操作以及输出信息；

（2）在日志查询模块输入要查询的日志时间端或者日志种类‘

（3）在日志分析模块输入要分析的日志信息；

## 7.5 输出项

（1）以分页形式向用户提交各种日志文件的查询信息；

（2）以分页形式向用户提交各种规则文件的查询与更新信息;

## 7.6 UI设计说明

## 7.7 流程逻辑

本系统共分为三个模块：显示模块，业务模块，业务处理模块；

(1) 显示模块用来给客户端用户提供一系列的页面视图，以接受客户端的访问。为客户端返回查询的内容，以及接受客户端输入的数据，并交给服务器进行处理；

(2) 业务模块负责将显示模块中涉及到的业务处理提交给相应的业务处理模块，处理完成后，将处理后的结果交给显示模块，其作用相当于一个控制器；

(3) 业务处理模块对业务中涉及到的请求和数据进行处理并进行持久性存储，包含了数据库的设计；

## 7.8 接口



图3-1日志文件与分析模块与其他模块接口

## 7.9 存储分配

## 7.10 注释设计

（1）对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

（2）对变量的注释：在该模块中程序用到的主要变量都作出必要的说明。

（3）对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明

（4）对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性

## 7.11 限制条件

## 7.12测试计划

## 7.13 尚未解决的问题

# 8.即时传输模块

## 8.1程序描述

本模块完成视频流的即使传输。辅助语义分割、物体检测模块的视频流传输显示。

通过python的opencv获取rtsp视频流，然后websocket实时发送base64图片到前端html显示。

## 8.2 功能

将程序输出的视频流显示至网页

## 8.3 性能

响应时间＜1秒

## 8.4 输入项

通过python的opencv获取rtsp视频流

## 8.5 输出项

html显示的base64图片

## 8.6 UI设计说明

无UI

## 8.7 流程逻辑

本系统共分为三个模块：显示模块，业务模块，业务处理模块；

(1) 显示模块用来给客户端用户提供一系列的页面视图，以接受客户端的访问。为客户端返回查询的内容，以及接受客户端输入的数据，并交给服务器进行处理；

(2) 业务模块负责将显示模块中涉及到的业务处理提交给相应的业务处理模块，处理完成后，将处理后的结果交给显示模块，其作用相当于一个控制器；

(3) 业务处理模块对业务中涉及到的请求和数据进行处理并进行持久性存储，包含了数据库的设计；

## 8.8 接口

公用后端函数get\_rtsp\_stream(); 获取视频流

## 8.9 存储分配

## 8.10 注释设计

（1）对程序的注释：在该模块中各程序的相关信息和对程序功能的说明。

（2）对变量的注释：在该模块中程序用到的主要变量都作出必要的说明。

（3）对函数的注释：在该模块中每个自定义函数的前面都有对该函数的功能说明，参数调用情况，返回值等的说明

（4）对关键算法的注释：对比较复杂的算法加以专门的说明，以便加强程序的可读性

## 8.11 限制条件

## 8.12测试计划

## 8.13 尚未解决的问题