

|  |
| --- |
| **金陵科技学院-南京绕越高速管理中心**  **交通事件智能检测系统** |

**项目代码文档**

**学校：东南大学**

**学院：交通学院**

**项目组成员：Feng R., Tian D.**

**指导老师：Li Z**

2024年3月

|  |
| --- |
| **File Status:**  [√] Draft  [ ] Released  [ ] Active  [ ] Expired |

|  |
| --- |
| **Proprietorship** |
| The document and the information contained are the property of the SEU Li Z. research group. The copy, use and disclosure of the document and its information must be authorized by SEU Li Z. research group in writing. |

**Revised State**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Version/**  **Revision Number** | **Content Modification** | **Modifier** | **Modification Date** |
| 1 | 0.1 | 初始草稿 | Tian Dongyu | 2024.3.22 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 项目概况 5](#_Toc161775624)

[1.1 项目单位 5](#_Toc161775625)

[1.2 项目需求 5](#_Toc161775626)

[1.3 项目时间 5](#_Toc161775627)

[1.4 工作安排 6](#_Toc161775628)

[2 项目分析 7](#_Toc161775629)

[2.1 研究内容 7](#_Toc161775630)

[2.2 事件检测类别 7](#_Toc161775631)

[3 代码开发规范 8](#_Toc161775632)

[3.1 书写规范 8](#_Toc161775633)

[3.2 开发环境 9](#_Toc161775634)

[3.3 开发架构 9](#_Toc161775635)

[4 数据协议 11](#_Toc161775636)

[4.1 接收数据 11](#_Toc161775637)

[4.2 上报event数据 13](#_Toc161775638)

[4.3 代码内targets数据 14](#_Toc161775639)

[4.4 代码内events数据 15](#_Toc161775640)

[5 测试数据概况 18](#_Toc161775641)

[5.1 轨迹点平面分布 18](#_Toc161775642)

[5.2 车道时空图 19](#_Toc161775643)

[5.3 雷达数据问题 20](#_Toc161775644)

[6 项目启动阶段 21](#_Toc161775645)

[6.1 项目规划与难度评估 21](#_Toc161775646)

[6.2 项目启动注意事项 22](#_Toc161775647)

[7 项目开发阶段 23](#_Toc161775648)

[7.1 代码编写标准 23](#_Toc161775649)

[7.2 大型项目代码策略 23](#_Toc161775650)

[7.3 工程代码策略 23](#_Toc161775651)

[7.4 代码测试 23](#_Toc161775652)

[7.5 算法性能评估 23](#_Toc161775653)

[8 项目联调阶段 24](#_Toc161775654)

[9 项目部署阶段 25](#_Toc161775655)

# 开发规则

## 书写规范

本项目代码书写规范采用python官方编写规范PEP 8 CODE CHECKER。

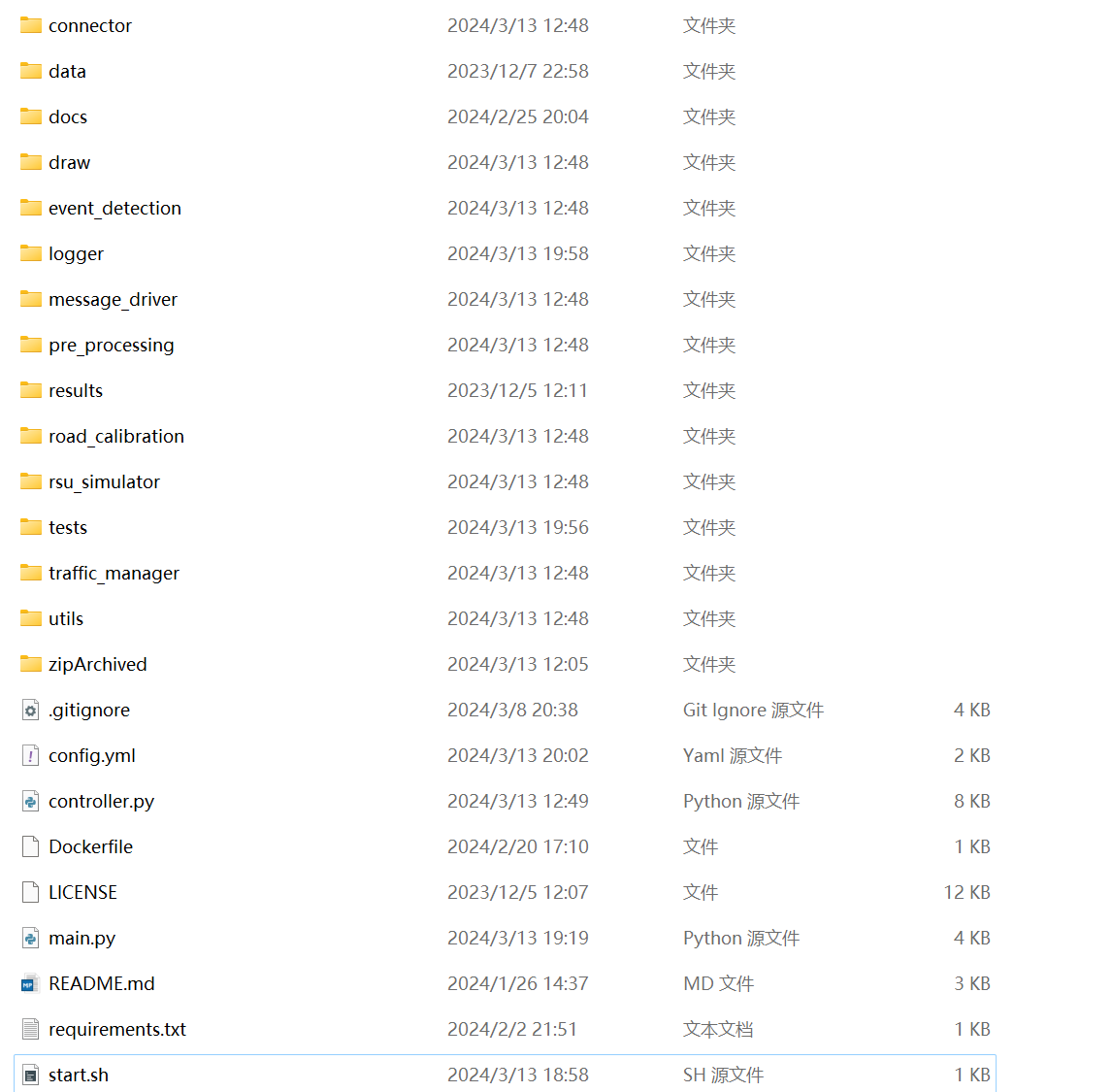
参见标准：<https://peps.python.org/pep-0008/>，代码上传到github需通过lint检查（flake8），审查命令：

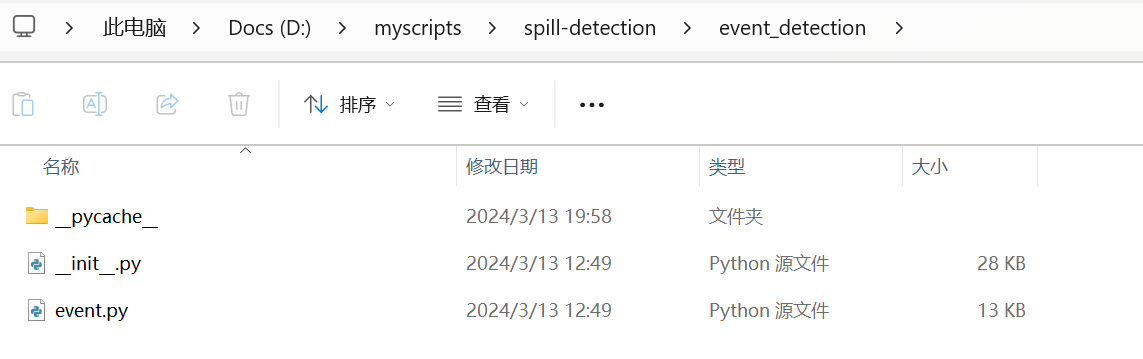
|  |
| --- |
| flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics  flake8 . --count --exit-zero --max-complexity=10 –statistics |

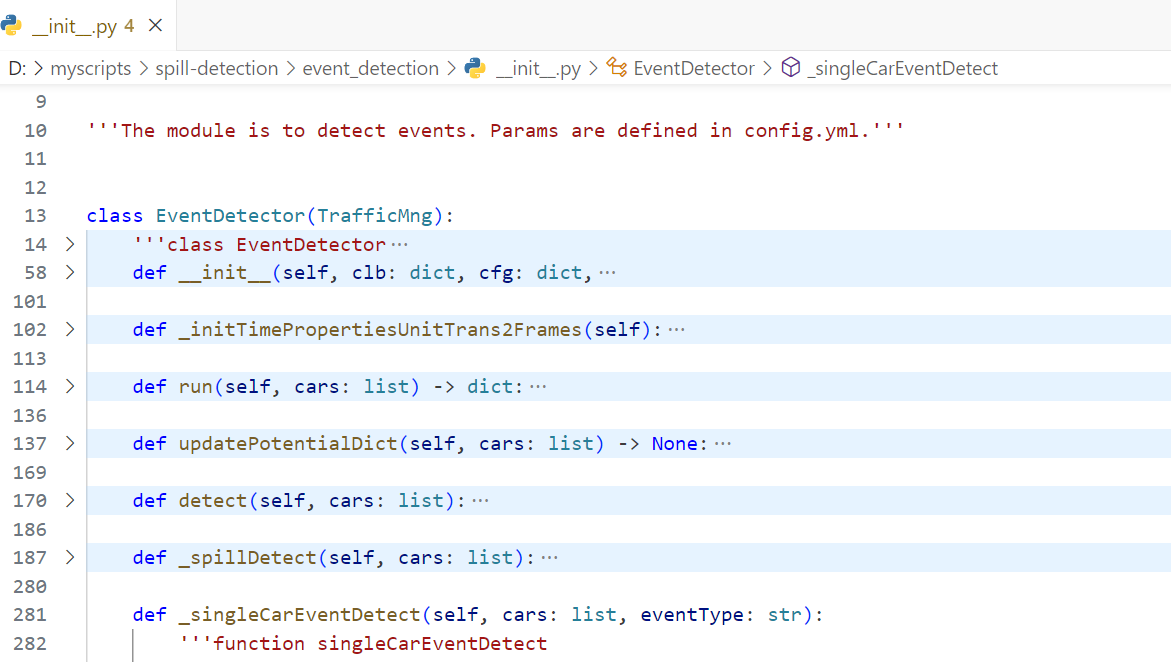
规范主要内容：

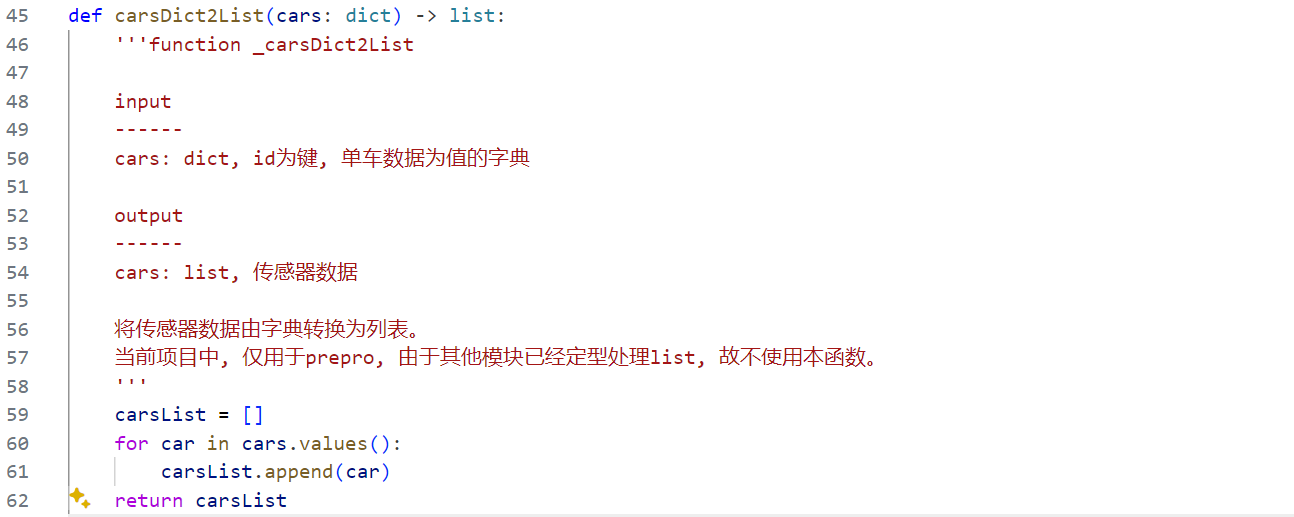
* 1. 变量命名采用驼峰标准（myFirstName、myLastName）；
  2. 函数分模块（文件夹）编写，各模块内设置主class, module接口平行，进出接口都一样,即各class应设置统一命名的外部调用函数（如run()）；
  3. class内的函数，通过前缀\_表明某函数不被外部调用，为私有函数；
  4. 所有code都需要写对应的单元测试，利用pytest检查；
  5. 开发过程随时更新文档，代码需要规范注释，参数返回要表明type，注释要写明input和return和功能。

规范情况见下页内容。









## 开发环境

所有开发调测全部基于Linux Docker环境，或者在windows开发，在linux/docker内完成测试。

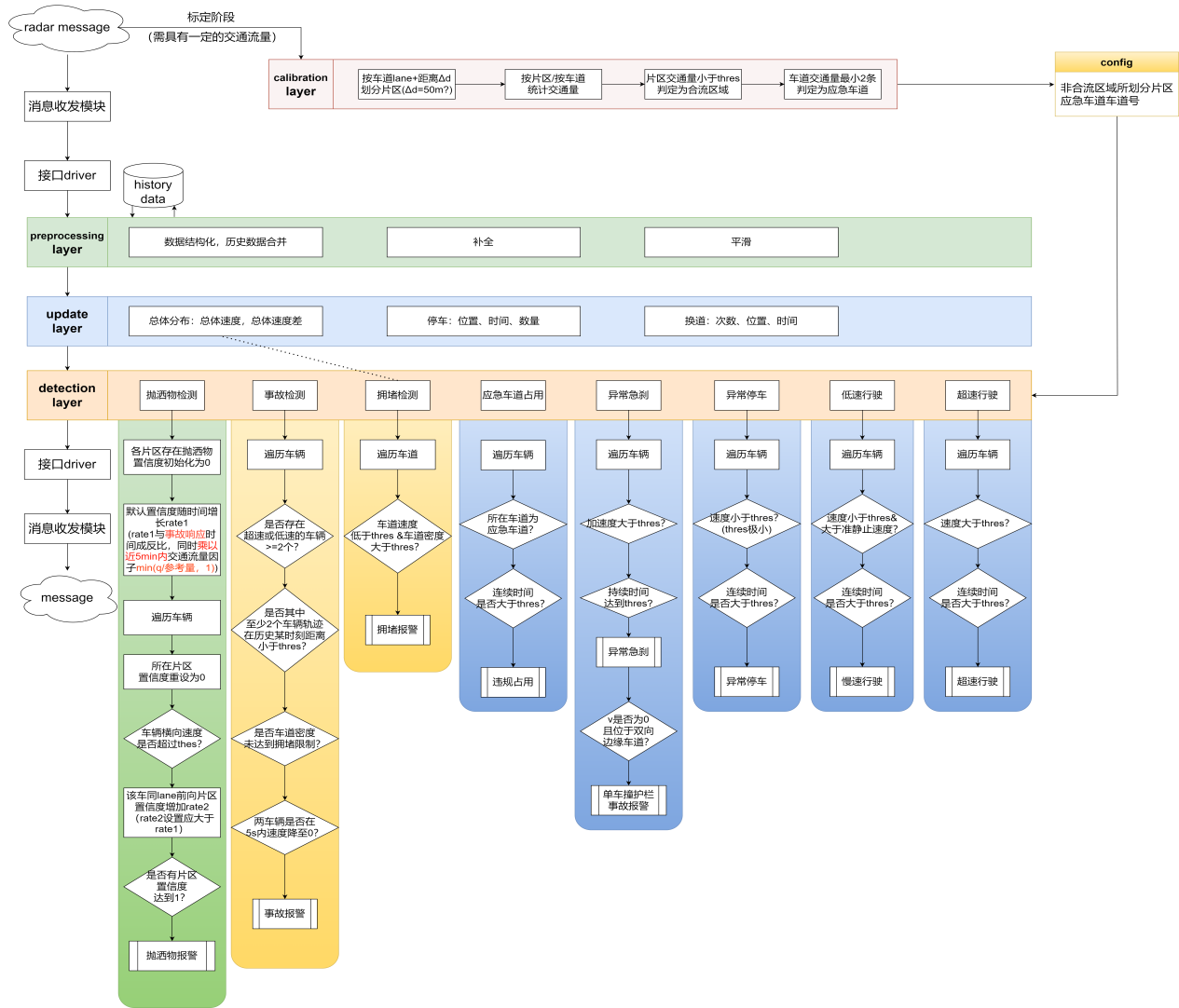
# 项目说明

## 项目目标

构建高速公路异常事件检测交通处理模型：采用运行**轨迹特征判断**、**交通行为判断**方法，对视频可观察到的**路面抛洒物事件**以及引起的**事故事件**进行检测；**对视频不可观察的异常条件下拥堵异常行为进行异常检测**。

需要提供算法源码1套，算法模型源码必须**开源且可定制接口**。

## 项目算法

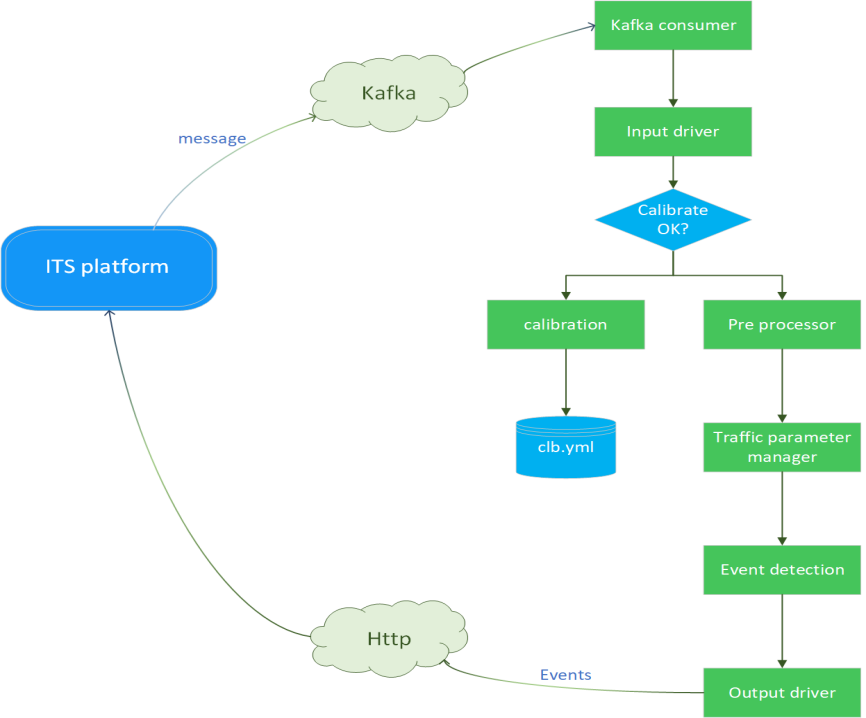


## 开发架构

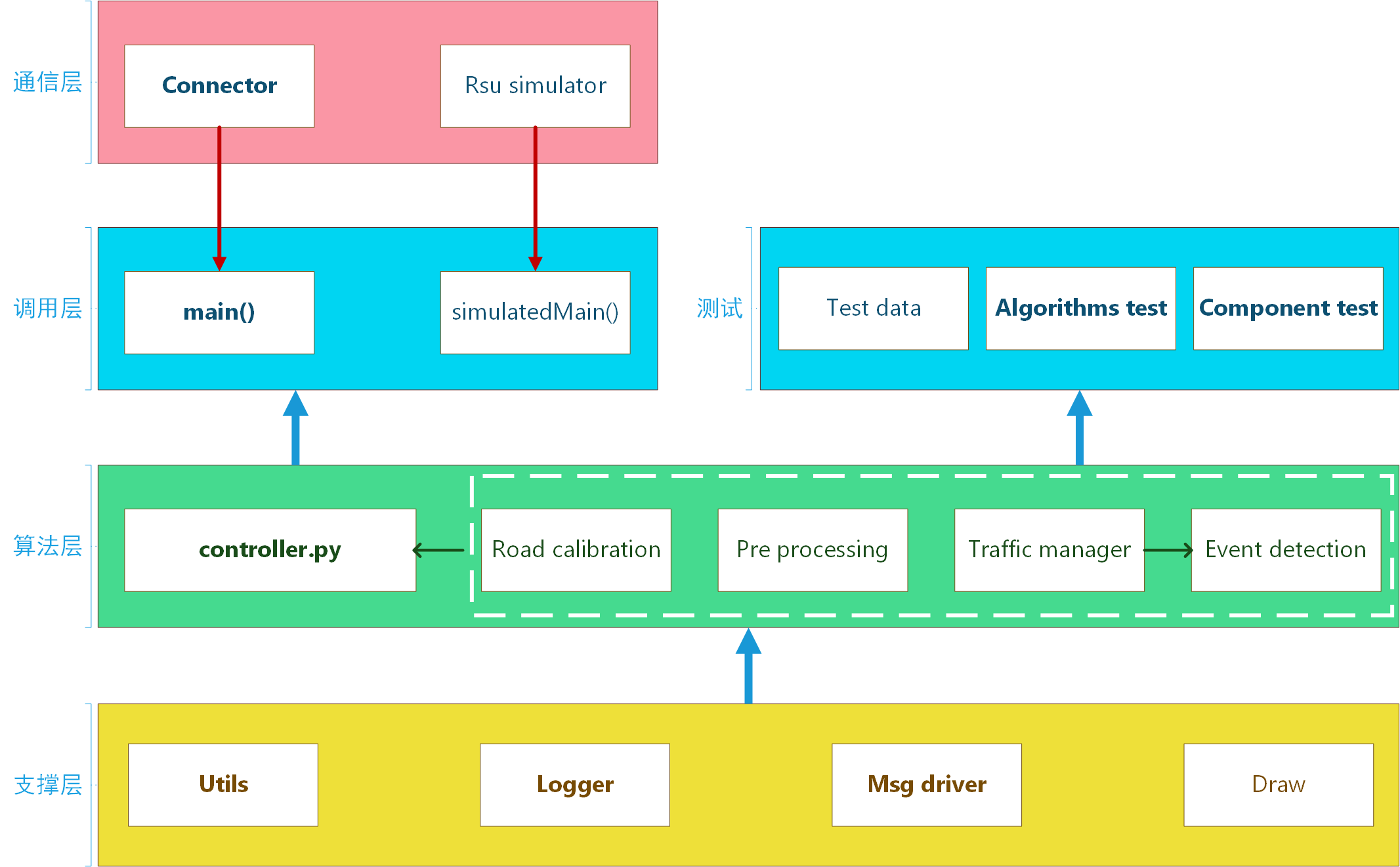
项目开发架构将能够拆解的独立流程独立为一个module，通过外部main.py实现调用。

|  |  |  |
| --- | --- | --- |
| **模块** | **功能** | **说明** |
| **controller** | **算法pipeline** | **算法主流程** |
| logger | 日志器 | 日志保存 |
| rsu-simulator | 数据传输模拟器 | 模拟实时数据按帧传输 |
| **connector** | **数据传输接口** | **kafka，http传输** |
| **driver** | **数据格式驱动** | **出入代码的数据格式转换** |
| **road-calibration** | **路段参数标定** | **适应多设备不同场景** |
| **pre-processing** | **数据预处理** | **处理问题轨迹数据** |
| **traffic-manager** | **交通参数管理器** | **记录更新路段交通参数** |
| **event-detection** | **事件检测** | **根据交通参数和轨迹检测事件** |
| utils | 通用函数 | 存储各模块常用函数 |
| tests | 测试 | 各模块测试函数 |
| **Assets** | | |
| docs | 存储项目文档 | 框架和算法必要说明 |
| data | 存储测试数据 |  |
| draw | 轨迹数据可视化 |  |

## 数据流通架构



## 模块依赖关系



# 支撑层模块

## 常用工具utils

存有整个项目常用的常量和函数。

### \_\_init\_\_.py

通用的基础工具。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| defaultConfig | 设定了配置的默认数值 | 配置缺省时调用 |
| loadConfig(path) | 读取config文件 | 缺省时调用默认配置 |
| loadYaml(path) | 读取yaml文件并返回 |  |
| updateDictCount(dic, key) | 更新计数dict数值 |  |
| delDictKeys(keys) | dict删除多个键 |  |
| **int2strID(num, length)** | 整数转为指定长度的id |  |
| strCapitalize(s) | 字符串首字母大写 |  |
| **unixMilliseconds2Datetime(**  **unix\_milliseconds)** | 将Unix毫秒时间戳转换为年月日时分秒格式字符串 |  |
| **isNotTargetDevice(msg, args)** | 是否不是目标设备 |  |
| **isInvalidMsg(msg)** | 是否是无效消息 |  |

### Car\_util.py

专门处理车辆数据的工具函数。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| getCarFromCars(cars, id) | 从传感器数据中获取指定id的单车数据 |  |
| carsList2Dict(cars) | 将传感器数据按id组织为字典 |  |
| carsDict2List(cars) | 将传感器数据由字典转换为列表 |  |
| **getCarBaseInfo(cars)** | 获取单车基本信息字符串 | 包括x,y, laneID, vx,vy。用于事件报警日志 |

### default.py

定义了默认的事件类别及相应标签。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| **defaultEventTypes** | 默认事件类别，8类。 | ["spill", "stop", "lowSpeed","highSpeed","emgcBrake","incident","crowd","illegalOccupation"] |
| **typeIdDict** | 默认8类事件对应的id | 0~7 |
| **typeCharDict** | 默认8类事件对应的大写字母 | A~H |

## 日志器logger

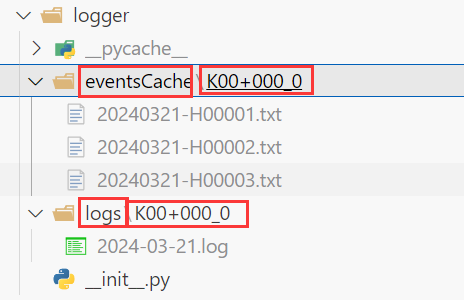
定义了日志器，并且在该文件夹下保存日志记录和事件发生时的缓存数据。

### 日志器\_\_init\_\_.py

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| **class** MyLogger(基于python标准库logger生成) | | |
| **updateDayLogFile()** | 新一天会更新日志保存到新的文件 | main中每帧执行，检查是否更新 |
| \_checkLogsDir() | 检查日志文件夹是否存在 | 日志文件夹包括logs和eventsCache, 该函数负责检查logs。 |

### 日志文件夹

两个文件夹均保存在对应设备号的子文件夹。Log按天保存，eventsCache按日期-事件id保存。



## 数据格式驱动器driver(message\_driver)

该module定义了数据格式驱动器，将外部传来的轨迹数据转化为代码内部处理的标准格式，并在上报事件时将事件数据转化为数据协议中要求的事件格式。

因项目中涉及的离线样本数据与实际部署数据有格式差异，因此定义了driverOnline和driverOffline，子函数一致，实现相同功能，但具体细节不同。同时定义了统一接口的driver供外部调用。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| **class** Driver() | | |
| **setLanes(lanes)** | 接收来自标定文件的lanes, 设为自身属性 | online使用，出现目标的lane不在setted车道中时，告警 |
| **receive(msg)** | 接受传来的数据message, 将原始数据格式转化为代码内流通的数据格式 | 根据字段内容判断online/offline并存为属性 |
| \_validMsg(msg) | 判断消息是否有效 | 各类无效数据剔除，主要防止空数据和字符串报文导致error |
| **send(msg)** | 将内部产生的events转化为协议所需事件格式 | 根据已存属性online/offline调用不同send |
| **class** DriverOffline() / DriverOnline() | | |
| **receive(msg)** | msg格式转化 |  |
| **send(cars, events)** | 数据转化为协议事件格式 |  |
| \_formatTransOuter2Inner(car) | 单目标的格式外转内 | receive调用 |
| \_formatTransInner2Outer(car) | 单目标的格式内转外 | send调用 |
| \_eventsInner2Outer(events) | 事件列表格式内转外 | send调用 |
| \_eventInner2Outer(event) | 单事件格式内转外 | send调用  仅online有 |

## 可视化draw

将轨迹数据等信息可视化。

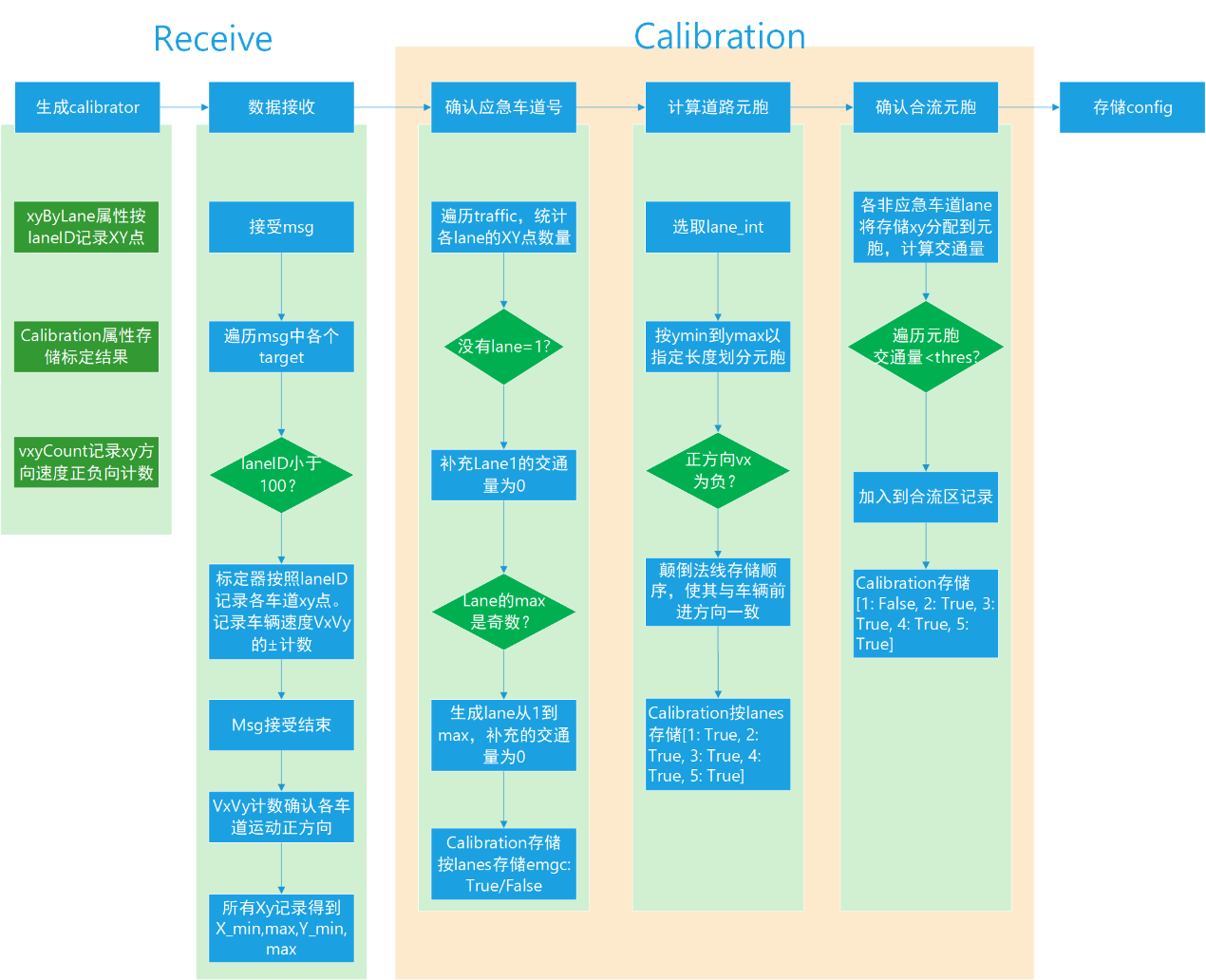
|  |  |  |
| --- | --- | --- |
| **函数** | **功能** | **说明** |
| trajXYPlot() | 简单画出xy散点图 |  |
| drawLanes() | xy散点图基础上画出车道 |  |
| drawVxVy() | 画出车辆xy与速度的分布 |  |
| drawClb() | 画出车道元胞标定结果 |  |

# 算法层模块

## 标定器calibrator(road\_calibration)

针对不同路段情况不同，可能有不同数量的车道、车道曲率不同（考虑到车道线的多项式拟合，需要保存车道线方程系数，但项目当前不考虑该问题）等状况，需要在接收某路段信息时，记录其必要信息。

### 标定算法流程



### Clb.yml标定信息

将保存在road\_calibration/clbymls文件夹下。

|  |
| --- |
| 1:    cells:    - false    - false    - true    - true    coef:    - -0.862    - -45.739    - 177.781    emgc: true    end: 0    len: 799.3    start: 799.3    vDir:      x: 1      y: -1  2:  …. |

### \_\_init\_\_.py

定义了标定器的主体函数。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class Calibrator | | |
| **run(msg)** | 接受每帧传输来的目标信息, 更新给calibrator | 缓存必要车辆点信息 |
| **calibrate()** | 标定一定时间后，结束标定流程，计算标定结果 |  |
| **save()** | 将标定结果保存为yml |  |
| \_distinguishNormalAndEmgcLanes() | 确定车道和应急车道列表 |  |
| \_distinguishInnerAndOutboardLaneID() | 确定内外部车道id | 因车道有一定弯曲，腹侧为内，背侧为外 |
| \_distinguishLaneDirection() | 确定各车道正方向 |  |
| \_calibXYMinMax() | 确定设备监测范围 |  |
| \_getLaneQuartilesPoints() | 获取各车道上车辆轨迹点的四分位点 | 按y坐标确定四分位特征点，用于车道方程的多项式拟合 |
| \_calculateLanesFunction() | 计算车道方程 |  |
| \_calibCells() | 标定各车道上各cell是否有效 | 标定出的cells有效否的顺序, 为沿着车道向前行驶的正方向 |

### algorithms.py

定义了路段标定涉及的算法函数。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| calQuartiles(points, range) | 计算一组数据点的四分位点, 采用四分位点附近范围内的点的均值。 | 将用于车道线拟合 |
| dbi(xys) | 给定单独一组数据点xy, 计算该组数据点在平面上的分散程度。 | \_distinguishInnerAndOutboardLaneID()调用，根据点的分散程度确定内外 |
| poly2fit(data) | 计算一组数据点的二次拟合曲线的系数, 系数分别代表x^2, x, 1的系数。 | 用于车道方程拟合 |
| poly2fitFrozen(data, a2) | 对一组数据点进行二次曲线拟合, 给定x^2的系数,  通过最小二乘法确定x系数和常数项, 返回系数为x^2, x, 1的系数。 |  |
| cutPts(start, end, len) | 给定起始位置、终止位置、元胞长度, 返回切割点。 | 若最后一个切割点到end的长度<len,从该位置继续增加len长度,不受到end限制。返回包括首尾。 |

## 预处理器preProcessor(pre\_process)

### \_\_init\_\_.py

定义了预处理器，是预处理的pipeline。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class PreProcessor | | |
| **run(curFrame)** | 进行补全, 平滑运算 | 该类所有函数回原地修改变量数值 |
| \_calAcceleration(contextFrames, curFrame) | 利用数据已有的vx与vy计算加速度a |  |
| \_fixSpeed0(curFrame) | 修正速度为0的数据 | 部分车辆速度值为0, 但vx和vy有数值 |
| \_copyTimestamp(cars) | 将车辆的timestamp属性备份出time属性 | 预处理算法会修改时间戳，应当为后续处理保留原始数值 |
| \_recoverTimestamp(cars) | 还原timestamp数值 |  |

### Utils.py

定义了预处理模块的帧数据处理函数。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| frameDelete(contextFrames, lastTimestamp) | 删除同一id过老旧数据、过久没有更新过的guid所有数据。 | 进行原地修改 |
| framesCombination(contextFrames,currentFrame ,lastTimestamp) | 将当前帧数据添加到历史帧数据中, 同时重置时间戳, 保证时间戳恒增 |  |
| getCurrentFrame(frames, lastTimestamp) | 从历史帧数据中提取出当前帧数据 |  |

### Complete.py

定义了插值法补全器。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class Interpolation | | |
| **run(contextFrames,currentFrame,lastTimestamp)** | 插值法补全车辆轨迹点 |  |
| \_findNearest(array, value) | 找出需要做补全的指定帧号下指定id的下标 |  |
| \_isFrameValid(objInfo, index, delaySecMark) | 根据帧时间戳判定该帧数据是否有效 |  |
| \_complete\_obj(objInfo, index, delaySecMark) | 补全指定的帧号下指定 id的轨迹点 |  |
| \_findDelaySecMark(frames) | 找到delaySecMark,并更新原frames的secmark |  |
| \_handleInterpolation(frames, delaySecMark) | 判断是否需要做补全, 并调相应函数做补全处理 |  |

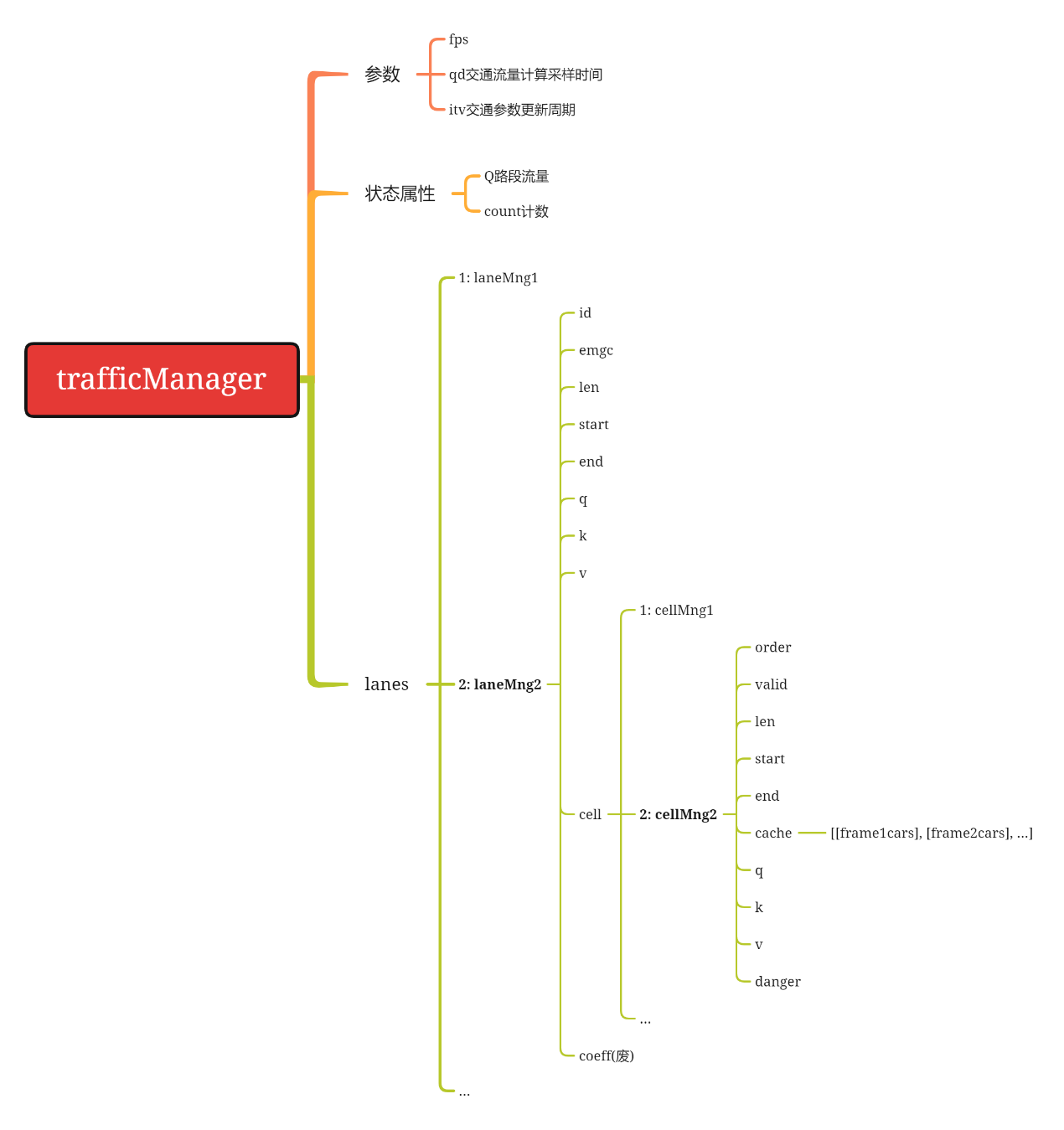
### Smooth.py

定义了指数平滑法平滑器。

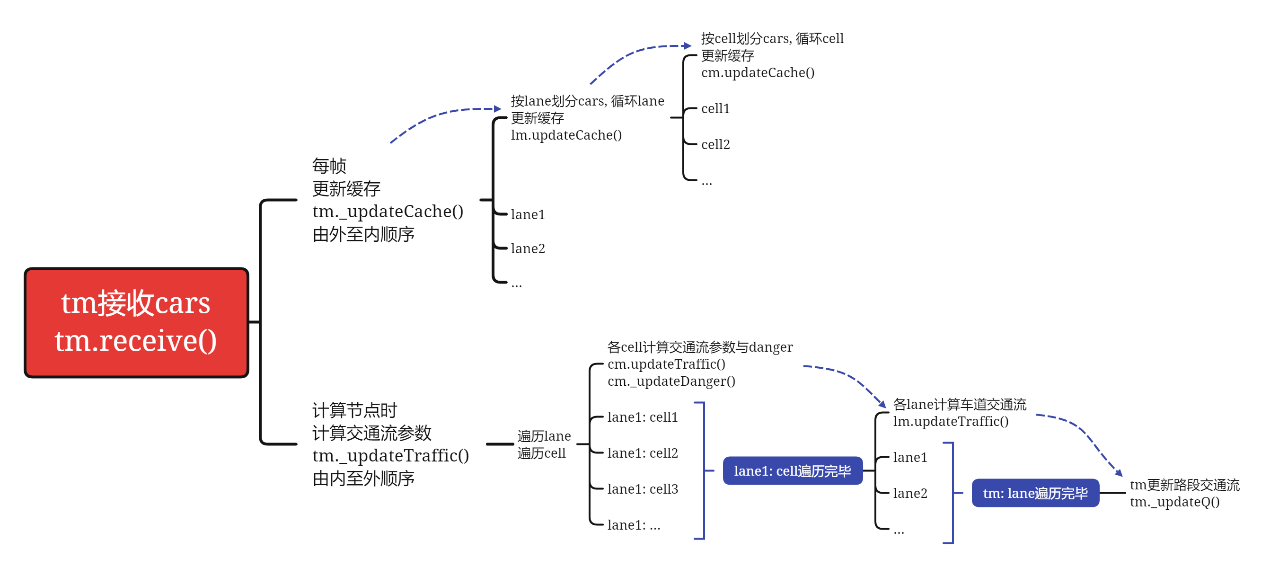
|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class Exponential | | |
| **run(contextFrames,currentFrame,lastTimestamp)** | 通过指数平滑函数对最新轨迹点进行平滑 |  |
| \_smooth\_one(objInfo, latestIdSet) | 调用一次指数平滑 |  |

## 交通参数管理器trafficManager(traffic\_manager)

### 交通数据管理架构



### 交通参数管理流程



### \_\_init\_\_.py

定义了整个路段的交通参数管理器trafficManager。

因其上层构建了event\_detection，对外接口函数若为run则会被覆盖，故对外接口函数写为update。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class TrafficMng | | |
| \_\_init\_\_(clb, cfg) | 初始化各级管理器 |  |
| \_initLanes(clb, cfg) | 初始化车道管理器 | \_\_init\_\_调用 |
| **update(cars)** | 更新交通参数 |  |
| **updateDanger()** | 更新元胞抛洒物危险度 |  |
| **resetCellDetermineStatus()** | 重置元胞的数值增加状态 |  |
| \_updateCache() | 更新缓存数据 | update()调用 |
| \_updateTraffic() | 更新计算交通流参数 | 指定频率由update()调用 |
| \_updateQ() | 更新计算路段交通流量 | \_updateTraffic()调用 |
| \_updateR1() | 将路段q传递给cell, 用于更新cell抛洒物置信度随时间增长的rate1 | 指定频率由update()调用 |
| \_carsByLane() | 按车道组织车辆 |  |

### Lane\_manager.py

定义了单个车道的交通管理器laneManager。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class LaneMng | | |
| \_\_init\_\_(\*) | 初始化车道管理器 |  |
| \_initCells(cfg) | 初始化元胞管理器 |  |
| **updateCache()** | 更新车道元胞缓存 | 被trafficMng同名函数调用 |
| **updateTraffic()** | 更新车道交通流参数 | 被trafficMng同名函数调用 |
| **updateR1()** | 将路段q传递给cell, 用于更新cell抛洒物置信度随时间增长的rate1 | 被trafficMng同名函数调用 |
| **updateDanger()** | 更新元胞抛洒物危险度 | 被trafficMng同名函数调用 |
| **resetCellDetermineStatus()** | 重置元胞的数值增加状态 | 被trafficMng同名函数调用 |
| \_carsByCell(cars) | 按元胞组织车辆 |  |
| \_carLocCell(car) | 根据车道的start, end, Len, 确定车辆所在元胞 |  |

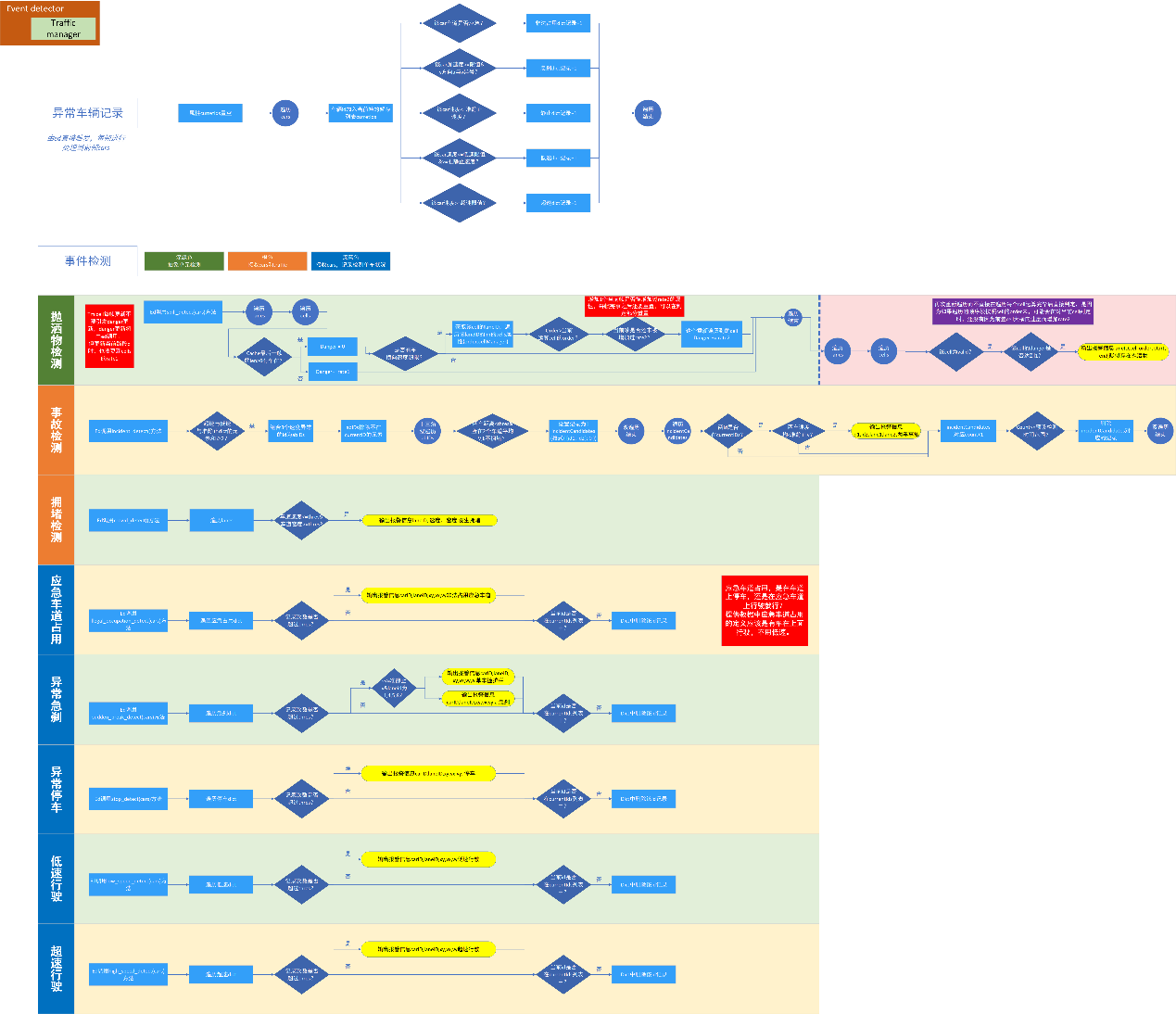
### Cell\_manager.py

定义了道路元胞交通管理器cellManager。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class CellMng | | |
| \_\_init\_\_(\*) | 初始化元胞管理器 |  |
| **updateCache()** | 更新车道元胞缓存 | 被laneMng同名函数调用 |
| **updateTraffic()** | 更新车道交通流参数 | 被laneMng同名函数调用 |
| **updateR1()** | 将路段q传递给cell, 用于更新cell抛洒物置信度随时间增长的rate1 | 被laneMng同名函数调用 |
| **updateDanger()** | 更新元胞抛洒物危险度 | 被laneMng同名函数调用 |
| updateDangerPassive() | 更新因较大横向速度导致的抛洒物危险度增长 | 被laneMng的**updateDanger()**调用 |
| **resetCellDetermineStatus()** | 重置元胞的数值增加状态 | 被laneMng同名函数调用 |

## 事件检测器eventDetector(event\_detection)

### 事件检测流程



### \_\_init\_\_.py

定义了事件检测器eventDetector，该类基于trafficManager生成。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class EventDetector | | |
| \_\_init\_\_(\*) | 初始化 |  |
| \_initTimePropertiesUnitTrans2Frames() | 初始化时间属性单位转换为帧 | 因采用timestamp而弃用 |
| **run(cars)** | 更新交通流信息, 检测交通事件, 输出并返回事件列表。 |  |
| **updatePotentialDict()** | 更新潜在事件记录变量, 将对应车辆的id和持续帧数记录在字典中 | 潜在事件包括: 静止, 低速, 超速, 急刹车, 非法占道。  负责上报事件结束。 |
| **detect(cars)** | 根据cfg中类别检测事件 |  |
| \_spillDetect(cars) | 检测抛洒物事件, 输出并返回事件列表 |  |
| \_singleCarEventDetect(cars) | 检测单车事件, 输出并返回事件列表 | 静止, 低速, 超速, 急刹车, 非法占道。 |
| \_stopDetect(cars)  \_lowSpeedDetect(cars)  \_highSpeedDetect(cars)  \_emgcBrakeDetect(cars)  \_illegalOccupationDetect(cars) | 检测5类单车事件，对外接口 | 调用\_singleCarEventDetect |
| \_incidentDetect(cars) | 检测多车事故事件, 输出并返回事件列表 |  |
| \_updateIncidentDict(cars) | 更新事故监测记录字典 | 被\_incidentDetect调用 |
| \_crowdDetect(cars) | 检测拥堵事件 |  |
| \_isCarStop(car)  \_isCarLowSpeed(car)  \_isCarHighSpeed(car)  \_isCarEmgcBrake(car)  \_isCarIllegalOccupation(car) | 判定单车在某一帧是否达到了某一事件的条件 | 返回bool |
| \_deleteNoUsePotentialDictKeys(type, cars) | 删除无效dict键, 并报警事件结束 | 被updatePotentialDict调用  包括当前帧丢失目标,或未消失但已脱离事件条件的目标 |

### Event.py

定义了事件管理器的格式，以及各个类型事件的事件格式对象。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class EventMng | | |
| \_\_init\_\_() | 初始化 |  |
| **run(type, startTime, endTime, \*info)** | 根据指定信息生成事件对象 | info可为car，lane，cell等实例 |
| clear() | 清空events, 在每帧事件检测结束后调用 | 由run调用 |
| \_generateEvent(type, eventID, startTime, endTime, info) | 根据type生成不同事件实例 |  |
| class BaseEvent  属性：type, eventID, startTime, endTime | | |
| class SpillEvent(BaseEvent)  class SingleCarEvent(BaseEvent)  class IncidentEvent(BaseEvent)  class CrowdEvent(BaseEvent) | | |
| \_\_init\_\_(\*) | 根据指定信息生成事件对象 |  |

# 通信层模块

## 数据传输器connector

定义了数据传输的http poster。Kafka直接调用了kafka包。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class HttpPoster | | |
| \_\_init\_\_(url) | 初始化 |  |
| **run(events)** | 按照协议要求, 逐个上报 |  |
| postData(event) | 将字典格式的数据以POST形式上传给http |  |

## 实时数据模拟器rsuSimulator

在离线条件下模拟实时的数据传输。方式为将数据读入内存，而后逐个调用。

|  |  |  |
| --- | --- | --- |
| **函数/常量** | **功能** | **说明** |
| class Smltor | | |
| \_\_init\_\_(dataPath) | 初始化 | 读取txt文件 |
| **run()** | 每运行一次run()函数, 读取一行数据 |  |

# 测试函数

## 测试数据benchmark

|  |  |
| --- | --- |
| **常量** | **测试用途** |
| dataNeedComplete | 补全测试用数据 |
| event | driver测试用事件数据 |
| eventData | 单个事件测试的汇总接口 |
| stadardClb | 标定测试用数据 |
| eachEventData(8组，包括input和output) | 8类事件的标准测试数据和事件结果，由eventData汇总接口 |

## 基础算法测试

测试基础组件的基础算法，保证基础无误。

|  |  |  |
| --- | --- | --- |
| **用途** | **函数** | **说明** |
| 标定 | test\_polyfit2 | 测试结果为预期数值 |
| test\_polyfit2A0 |
| cell定位 | testCarLocCell |
| 预处理 | test\_smooth |
| test\_complement |

## 功能测试

|  |  |  |
| --- | --- | --- |
| **用途** | **函数** | **说明** |
| 仿真器 | test\_simulator | 测试数据读取类别无误 |
| 驱动器 | test\_driver | 测试格式转化正确 |
| 标定器 | test\_calibrator | 测试标定结果为预期数值 |
| 预处理器 | test\_preprocess | 测试预处理后变量类别 |
| 交通管理器 | test\_manager | 测试交通量为预期数值 |
| 事件检测器 | test\_detect | 测试事件为预期数值 |

# 代码运行

## Main.py运行

运行：**python main.py –deviceId “K00+000” – deviceType 1**

|  |  |  |
| --- | --- | --- |
| **关键词** | **说明** | **数值** |
| deviceId | 设备编号/名称（常见为桩号） | str, 常见格式类似于K00+000 |
| deviceType | 设备类别 | int，雷达为1，视频为2 |

## 多设备start.sh运行

多设备配置在**start.sh中按照格式修改即可**。（利用linux系统的vim修改，避免windows修改sh文件）

