**系统详细设计说明书**

题目：公共地点人流量计算的云监管平台

学 院 计算机与电子信息学院

专 业 计算机科学与技术

班 级 计科162班

组 名 就来看看

组 长 黄健

成 员 雪雯 黄淑茵

指导老师 陈宁江

2019 年 5 月 1日

**目录**

[引言 4](#_Toc11092803)

[编写目的 4](#_Toc11092804)

[背景 4](#_Toc11092805)

[一、系统需求分析 4](#_Toc11092806)

[1.1功能性需求 4](#_Toc11092807)

[1.2非功能性需求 5](#_Toc11092808)

[二、系统功能说明 5](#_Toc11092809)

[2.1云平台功能描述 5](#_Toc11092810)

[2.2边缘端功能描述 6](#_Toc11092811)

[三、系统设计 7](#_Toc11092812)

[3.1逻辑结构设计 7](#_Toc11092813)

[3.2边缘端 7](#_Toc11092814)

[四、系统功能实现 8](#_Toc11092815)

[4.1 开发环境 8](#_Toc11092816)

[4.1.1云平台开发技术 8](#_Toc11092817)

[4.1.1边缘端开发技术 9](#_Toc11092818)

[4.2云平台监控端架构图 9](#_Toc11092819)

[4.3 云平台监控端数据流图 10](#_Toc11092820)

[4.4边缘端架构图 12](#_Toc11092821)

[4.5 边缘端数据流图 13](#_Toc11092822)

[4.6功能界面实现 14](#_Toc11092823)

[4.6.1 预警信息 14](#_Toc11092824)

[4.6.2 异常地点排序 15](#_Toc11092825)

[4.6.3 异常视频查看 15](#_Toc11092826)

[4.6.4 监控阈值设置 15](#_Toc11092827)

[4.6.5 选择处理视频 16](#_Toc11092828)

[4.7 前后端通信： 16](#_Toc11092829)

[4.7.1 HTTP API： 16](#_Toc11092830)

[4.8 视频识别处理模块 17](#_Toc11092831)

[4.8.1 OPENCV 17](#_Toc11092832)

[4.8.2 YoloV3 18](#_Toc11092833)

[4.8.3 Deep\_sort 19](#_Toc11092834)

[附录： 21](#_Toc11092835)

# 引言

## 编写目的

本详细设计说明书是针对公共地点人流量计算的云监管平台而编写的。主要用于为实现的功能进行的系统详细设计说明，详细的说明了系统各软件组成模块的实现流程、功能、接口、编译、测试要点等内容，进一步明确系统结构。本文档供小组全体成员及大赛评委阅读。

## 背景

无论是大型商超、交通枢纽等公共设施人流监测都为决策者提供资源分配合理化的理论支持，政府、企业和学校都需要云边协同的人流量计算云监管平台实现对人流密集度进行分析，对人群爆发聚集地点进行快速预警和疏导处理。

# 一、系统需求分析

## 1.1功能性需求

1、正确识3个以上人数密集边家区域的地点位置信息和人流量(即人头数) ,并做相应排序:

2、要求边缘端程序和云监营平台两个程序要求分开独立部署满足边缘计算云边协同架构的实际需求

3、边缘端程序识别人头数和地点信息并做相关计算，如果人头数没有达到容量上限则数据不上报到云端，若人头数超出预置容量上限数，就将所在人流异常视频数据地点信息和人头数上报到云监管平台

4、云监管平台呈现对应地点和人头数爆发的数据告警，边缘端人流数据实时变化，云端展示人流量数据也实时变化

5、异常视频中用动态标注的方式标注人头序号并识别地点信息和人头总数和警告

6、异常视频数据，地点信息和人头数数据在云端进行数据存储云监管平台可以查询

7、对多个异常地点进行数据分析，并分类，排序重点突出经常容易出现人流异常爆发的地点

8、云监管平台实时展示地点信息和人头数数据

9、智能匹配给出疏导处理的相关保安组织人员推荐信息和联系方式

10、web端体验良好

## 1.2非功能性需求

1、云监管平台用户体验好，流程易用；

2、识别算法的时间和空间复杂度最优，识别准确率高；

3、数据展示和传输无卡顿。

# 二、系统功能说明

云平台操作人员可以处理查看预警信息、序重点突出经常容易出现人流异常爆发的地点、查看指定地点的异常视频、设置监控地点的阈值、调用边缘端识别模块识别视频

## 2.1云平台功能描述

（1）预警信息

1、能在预警信息界面点击异常处理完毕按钮，将警告信息从界面移除，并向云平台发送HTTP请求，修改是否已经处理异常表的字段is\_deal = True,表示异常已经处理完成

2、不断发送ajax短轮询云平台，查看是否有新的异常信息，若有则在浏览器界面推送警告信息，并且显示该异常信息

（2）异常地点排序

向云平台发送HTTP请求，请求所有监控地点的信息（监控地址名、异常发生次数），而后云平台可以排序重点突出经常容易出现人流异常爆发的地点

（3）异常视频查看

指定监控地点，向云平台发送HTTP请求，获取该地所有异常视频，并且显示地点、异常人数、时间、视频(可以播放查看)

（4）监控阈值设置

指定监控地点，向云平台发送边缘端发送HTTP请求，设置监控该监控地点的阈值

（5）选择处理的视频

可以选择一个视频，向边缘端发送HTTP请求，边缘端会启动视频识别处理模块来处理该视频。而边缘端接收云平台发来的HTTP请求，根据请求的参数来决定对哪个测试视频进行识别，识别过程中，若有第一次超过阈值的情况则向云平台发送预警信息，而后开始不断推送接下来视频里的每时刻的人数。待识别结束后，将标记处理好的异常视频发送给云平台，云平台可以查看

## 2.2边缘端功能描述

（1）设置阈值

接收云平台发来的HTTP请求，根据请求的参数来设置指定监控地点的阈值

（2）视频识别处理

接收云平台发来的HTTP请求，根据请求的参数来决定对哪个测试视频进行识别，识别过程中，若有第一次超过阈值的情况则向云平台发送预警信息，而后开始不断推送接下来视频里的每时刻的人数。待识别结束后，将标记处理好的异常视频发送给云平台

# 三、系统设计

## 3.1逻辑结构设计

按照E-R图到逻辑关系模式的转换规则，可得到系统如下4个关系：

(1)Address

(地址ID、监控地址、异常发生次数、阈值);

(2)SecurityGuard

(保安ID、监控地址、姓名、电话);

(3)ExceptionsVideo

(视频ID、监控地址、人数、异常视频文件路径、异常到来时间)

(4)ToDoExceptions

(TID、监控地址、人数、保安名字、保安电话、视频ID、异常到来时间、异常是否处理完成);

# 四、系统功能实现

## 4.1 开发环境

### 4.1.1云平台开发技术

后端：

语言：python3.6

web框架：django-rest、django2.1

HTTP服务器：Nginx

数据库：MySQL

前端：

Js框架：vue.js

前端路由：vue-router

基础模板：vue-admin-template

Web UI框架：element-ui

HTTP libary: Axios

打包工具: vue cli3

包管理工具：npm

其他技术：html5、css3

IDE：pycharm

### 4.1.1边缘端开发技术

后端：

语言：python3.6

web框架：django2.1

HTTP服务器：Nginx

数据库：SQLite

多物体识别跟踪：OPENCV、YOLOV3、DEEP\_SORT（Kalman-filter、 Hungarian Algorithm）

## 4.2云平台监控端架构图

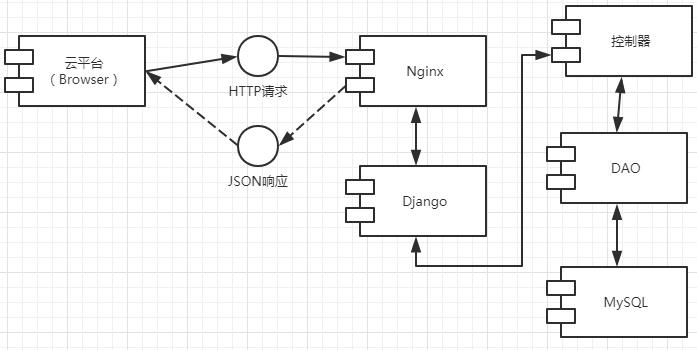


图1

## 4.3 云平台监控端数据流图

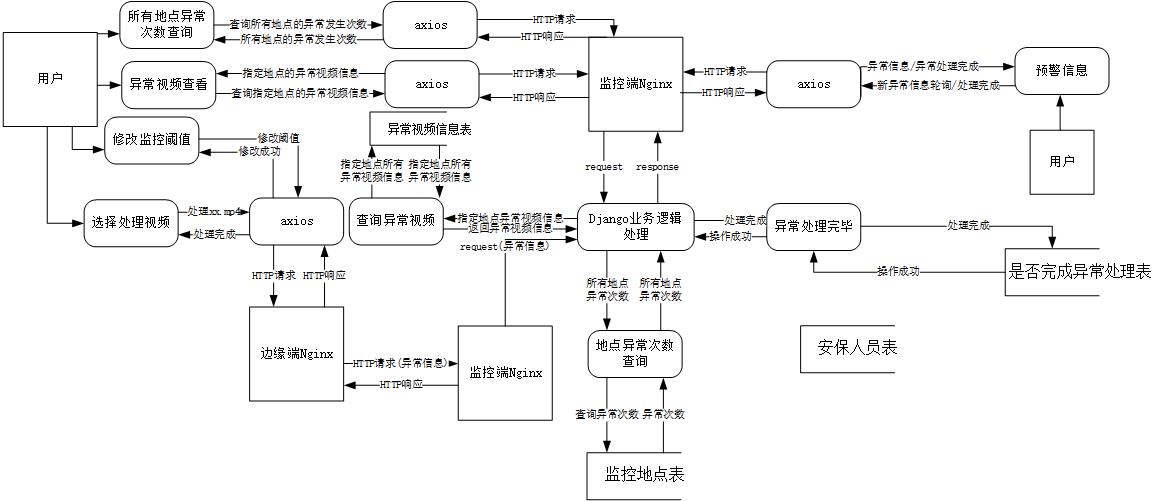
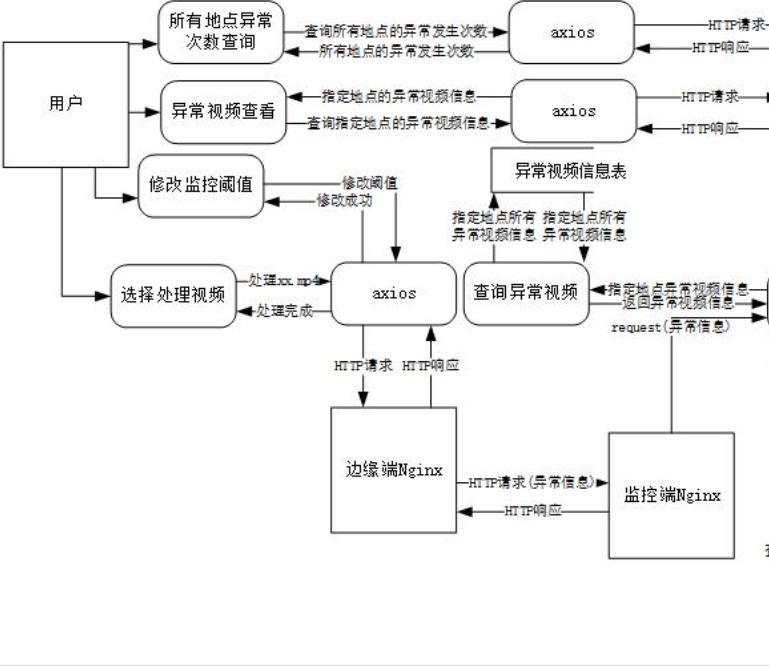


图2

由于图2数据流图过长，分成了两截方便查看，如下：

图3（1）

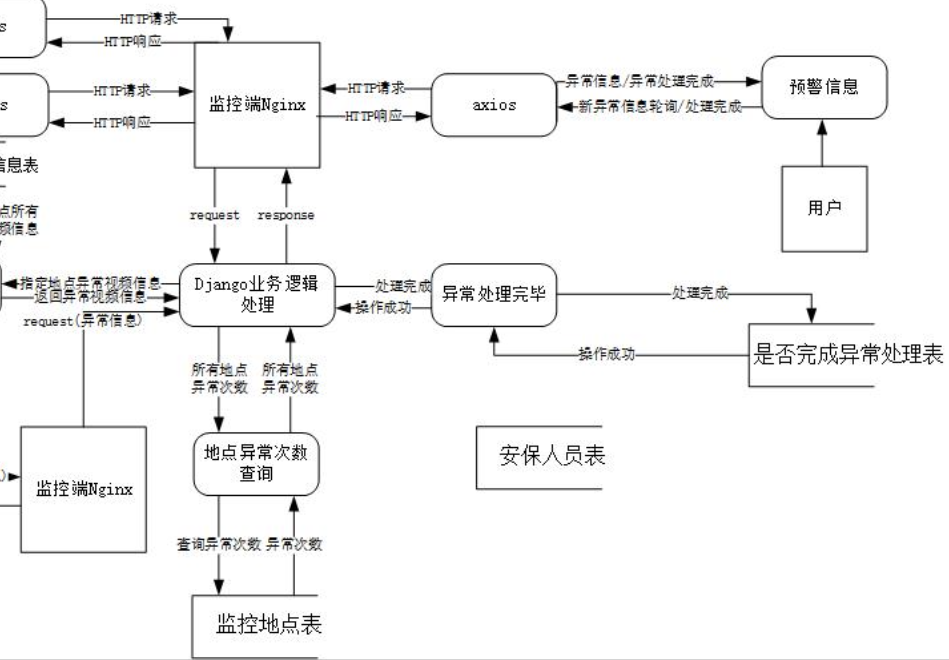


图3（2）

## 4.4边缘端架构图

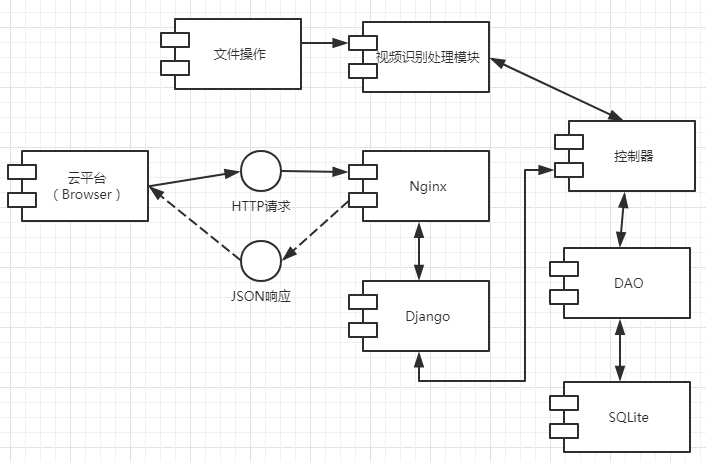


图4

## 4.5 边缘端数据流图

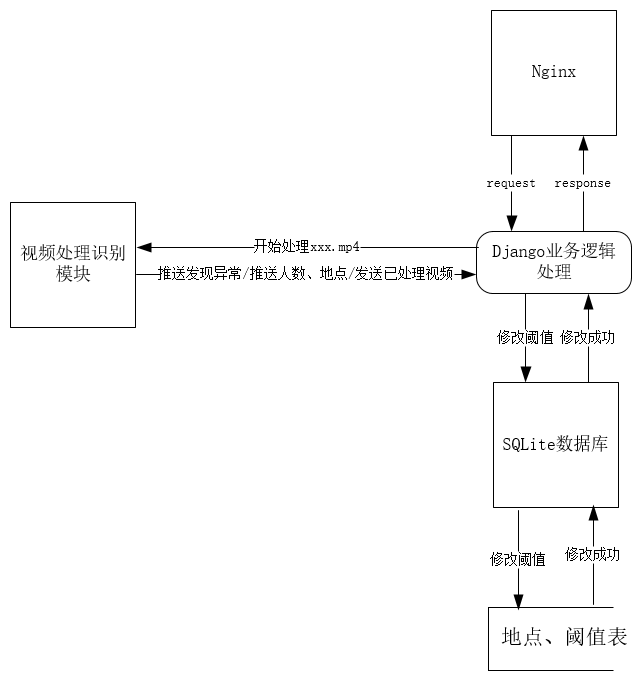


图5

## 4.6功能界面实现

### 4.6.1 预警信息

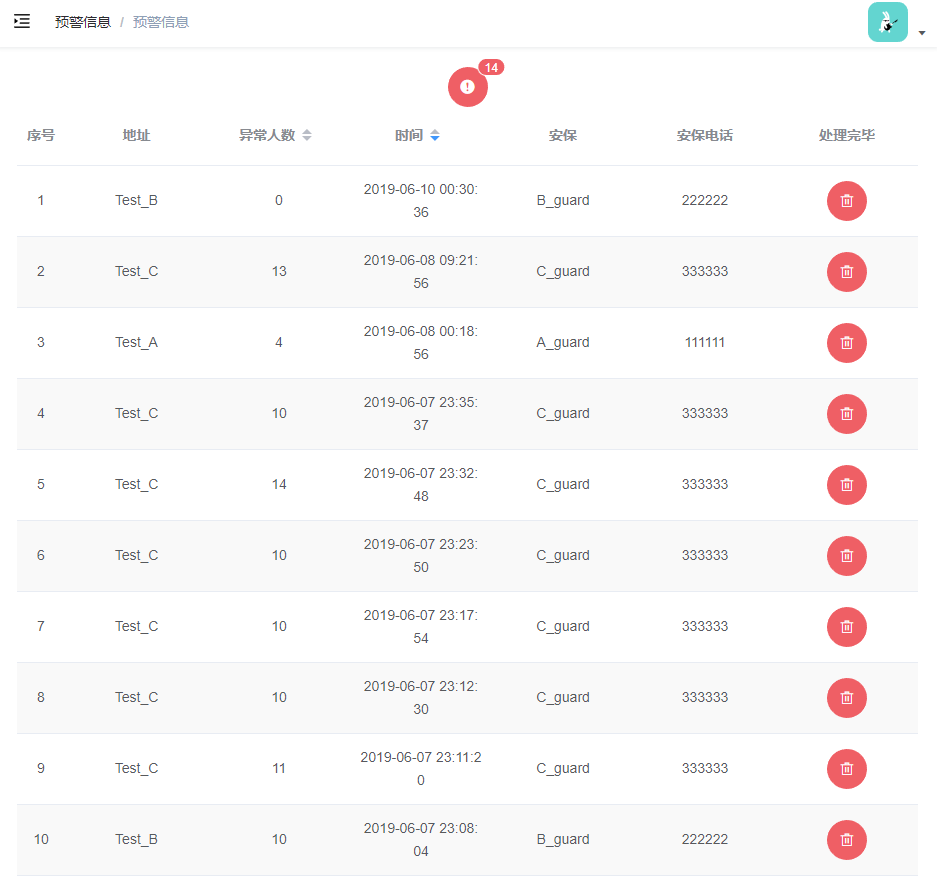


图6

### 4.6.2 异常地点排序

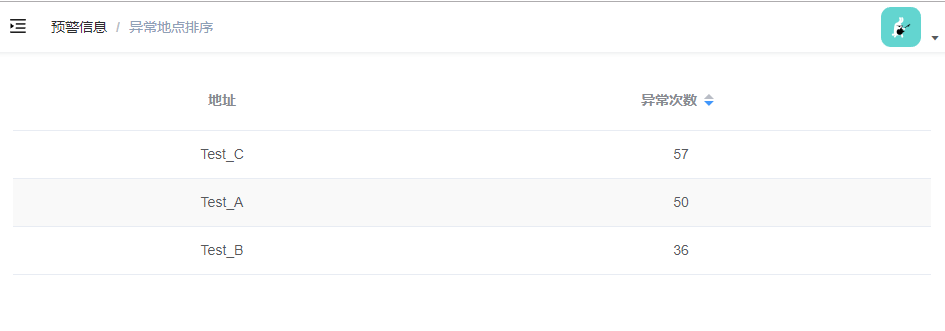


图7

### 4.6.3 异常视频查看



图8

### 4.6.4 监控阈值设置

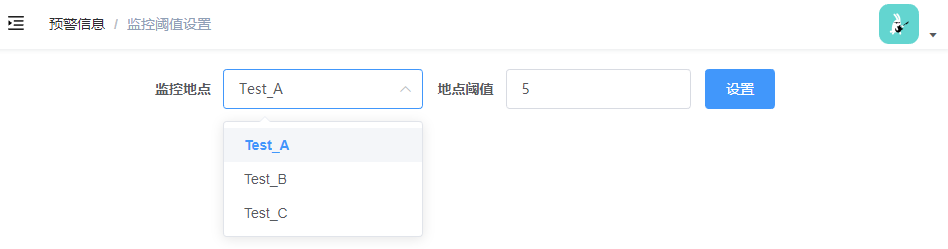


图9

### 4.6.5 选择处理视频



图10

## 4.7 前后端通信：

### 4.7.1 HTTP API：

**预警信息**

获取未处理的异常信息：GET '/to\_do/';

是否有新的预警：GET `/is\_new/`

已处理一个异常：GET `/is\_deal\_exception/`

**异常地点排序**

获取地点与异常次数：GET `/exceptions\_info/`

**异常视频查询**

获取指定地点异常：GET `/video\_info/?address=X`

**监控阈值设置**

从边缘端获取视频与阈值：GET `http://127.0.0.1:8000/address\_list/`

修改指定地点边缘端阈值：PUT `http://127.0.0.1:8000/update\_threshold/`

**选择处理视频**

已处理一个异常：post `http://127.0.0.1:8000/start\_recognize/`

{“video\_list”:[{

'address': 'Test\_C',

'video\_path': 'edgeapp/Test\_Video/t3.mp4'

}]

}

## 4.8 视频识别处理模块

### 4.8.1 OPENCV

1、视频读取

将指定路径或摄像头的视频读取到内存中，方便后续对视频做操作

2、视频操作

读取出视频的一帧，作为一张图片，并使用yolov3、deep\_sort处理这张图片，得出这一帧图片人物的标记框的位置，而后使用OPENCV将此框在这帧图片上绘制出来，并且有个标号。

3、生成视频

视频所有帧数处理完之后，生成一个视频，即：处理过的视频

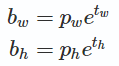
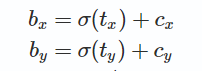
### 4.8.2 YoloV3

1、Bounding-Box预测

bbox 预测使用维度聚类（dimension clusters ）作为 anchor boxes 来预测边界框. 在训练期间，使用平方误差损失的总和。

2、Bounding-Box回归

使得第一步预测出来的Bounding-Box更接近于真实的标记，核心公式如下



3、多尺度预测

使用类FPN结构,使得模型可以对不同尺寸的图片、视频进行预测

4、CNN特征提取

对于所有的先验Bounding-Box，利用CNN提取feature map，并且层中间采用batch normalization来提升训练速度和最终的mAP。将feature map送入多个logistic分类器代替，来预测类别

5、多个logistic分类器

将feature map送入多个logistic分类器，对于所有先验Bounding-Box给出一个类别概率，然后给先验Bounding-Box打上类别标记与概率

6、OPENCV绘制框

OPENCV利用先验Bounding-Box的坐标，在原图绘制出标记框、类别、概率

7、只显示人

由于云监控平台的任务只要求识别人，所以对于所有先验Bounding-Box，只保留人的类别，并且我们对剩余的Bounding-Box做一个计数就得出了该帧图片里的人数

### 4.8.3 Deep\_sort

对于YOLOV3识别出的人的边框进行跟踪，并且赋予该边框一个ID，用以标记这个物体

1、预测模型Kalman-filter

使用一个8维空间去刻画轨迹在某时刻的状态：



使用一个kalman滤波器预测更新轨迹，该卡尔曼滤波器采用匀速模型和线性观测模型。通过卡尔曼估计对(u, v, r, h)进行估计，u，v是物体中心点的位置，r是长宽比，h是高。运动估计对于运动状态变化不是很剧烈和频繁的物体能取得比较好的效果。

其观测变量为:



2、轨迹处理

对于每一个追踪目标，都有一个阈值ak用于记录轨迹从上一次成功匹配到当前时刻的时间（即连续没有匹配的帧数），我们称之为轨迹。当该值大于提前设定的阈值Amax则认为该轨迹终止，直观上说就是长时间匹配不上的轨迹则认为该轨迹已经结束。

在匹配时，对于没有匹配成功的目标都认为可能产生新的轨迹。但由于这些检测结果可能是一些错误警告，所以对这种情形新生成的轨迹标注状态’tentative’，然后观查在接下来的连续若干帧（论文中是3帧）中是否连续匹配成功，是的话则认为是新轨迹产生，标注为’confirmed’，否则则认为是假性轨迹,状态标注为’deleted’。

3、分配问题的评价指标

在位置度量上，使用马氏距离（Mahalanobis distance）来评价卡尔曼滤波预测的状态和实际状态的匹配程度（运动匹配程度）:

4、级联匹配

如果一条轨迹被遮挡了一段较长的时间，那么在kalman滤波器的不断预测中就会导致概率弥散。那么假设现在有两条轨迹竞争同一个目标，那么那条遮挡时间长的往往得到马氏距离更小，使目标倾向于匹配给丢失时间更长的轨迹，但是直观上，该目标应该匹配给时间上最近的轨迹。

# 附录：

1、工程文件目录结构图

**云平台前端：**

Source Code/MonitorOfCloud/fronted

**云平台后端：**

Source Code/MonitorOfCloud/backend

**边缘端后端：**

Source Code/Edge